Fuzzy Control and Fuzzy Kinematic Mapping for a Redundant Space Robot

Alois Schacherbauer and Yangsheng Xu

CMU-RI-TR-92-12

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

August 1992

REPORT DOCUMENTATION PAGE

OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1992	technical

4. TITLE AND SUBTITLE

Fuzzy Control and Fuzzy Kinematic Mapping for a Redundant Space Robot

5. FUNDING NUMBERS

6. AUTHOR(S)

Alois Schacherbauer and Yangsheng Xu

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

8. PERFORMING ORGANIZATION REPORT NUMBER

CMU-RI-TR-92-12

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release;
Distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This report documents parts of the research in the Self Mobile Space Manipulator project at Carnegie Mellon University. We developed Fuzzy Logic Friction Compensation schemes that improve motion performance of SM2. Both static and dynamioc errors are reduced. Also, we propose Fuzzy Inverse Kinematic Mapping to resolve the redundancy problem in SM2. The proposed scheme works identically for redundant and non-redundant robots, does not require any constraints to be imposed on the robot configuration and provides a closed-form solution . We investigated this scheme in simulation and then implemented it for real-time teleoperation of SM2.

14. SUBJECT TERMS

Self Mobile Space Manipulator, fuzzy logic friction compensation schemes, real-time teleoperation

15. NUMBER OF PAGES

60 pp

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
unlimited	unlimited	unlimited	unlimited

Table of contents

List of Figures

List of Tables

Abstract

This report documents parts of the research in the Self Mobile Space Manipulator (SM^2) project at Carnegie Mellon University. We developed Fuzzy Logic Friction Compensation schemes that improve motion performance of SM^2. Both static and dynamic errors are reduced. Also, we propose Fuzzy Inverse Kinematic Mapping to resolve the redundancy problem in SM^2. The proposed scheme works identically for redundant and non-redundant robots, does not require any constraints to be imposed on the robot configuration and provides a closed-form solution. We investigated this scheme in simulation and then implemented it for real-time teleoperation of SM^2.

1.0 Introduction

1.1 Problem Statement and motivation for research

The Self Mobile Space Manipulator (SM^2) has been designed and built to work on Space Station Freedom. The robot is very light-weight, and thus highly flexible. Due to the light-weight structure and the zero-gravity-environment, the torques necessary to drive the robot are much lower than the torques required in industrial applications. Therefore, the relative torque necessary to overcome static friction is significant, in our application about 30% of the peak torque, compared to about 3% for normal industrial robots. Joint friction is difficult to model, because it is a function of configuration, payload and velocity, while simple friction compensation methods are not effective enough and also difficult to tune. Therefore, we have approached this issue by applying Fuzzy Logic friction compensation obtained directly from experiments.

The second problem is the fact that SM^2 is a redundant robot. Inverse Kinematics for redundant robots requires an explicit task model, and optimization function, and environmental model, and depends on numerical computation. This is a severe draw-back for real-time implementations, especially for teleoperation controlled robots, such as SM^2. Furthermore, many schemes display degenerated performance at singular positions.

We propose to use Fuzzy Logic for the Inverse Kinematic Mapping. It is independent of the number of links, results in zero steady-state-error and requires no optimum criterion or model. The usability of the approach has been demonstrated by implementing the proposed method for the real-time teleoperation of SM^2.

1.2 Fuzzy Logic and Fuzzy Control

A Fuzzy Logic Controller (FLC) is a controller that differs from a conventional controller (such as PID) mainly in that it works internally with Fuzzy variables. It comprises a knowledge base with definitions of membership functions and a rule-base, a decision-making logic, and interfaces to and from the physical world which allow the conversion from Fuzzy values into crisp values and vice versa.

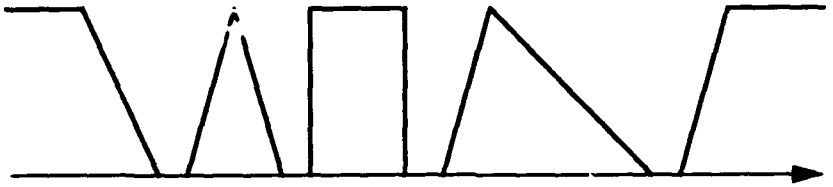
A control cycle typically consists of taking process variables as input, converting them to Fuzzy values, applying the input to the rule-base and deriving a Fuzzy control action, converting this Fuzzy control action to a crisp value, and giving this crisp value to the controlled process as control action. The following paragraphs explain Fuzzy variables and rule-base in more detail.

Fuzzy Variables are used to describe vague concepts such as **high** error or **little** oscillations. These concepts are defined by a membership function μ, which maps the values of the universe of discourse onto the interval of real numbers from zero to one:

$$\mu: \ u \rightarrow [0;1] \qquad \text{(EQ 1-1)}$$

The degree of membership $\mu_D(x)$ denotes how appropriate it is to represent x by the Fuzzy value D. The membership functions that we used in this project are piecewise linear functions, and samples of the available types are shown in Figure 1.

Figure 1 Implemented membership function types for SM^2 project.



Fuzzy rules are of the form

IF (*in1* = *u*) **AND** (*in2* = *v*) **THEN** (*out* = *w*),

where *in1*, *in2* and *out* represent Fuzzy variables, and ***u***, ***v*** and ***w*** are Fuzzy values.

In out project, the sentence connective 'AND' and the implication are based on the algebraic product. The weight α of a rule is hence determined by

$$\alpha = \mu_u(in1) \cdot \mu_v(in2). \qquad \text{(EQ 1-2)}$$

The defuzzified output of the FLC is derived as

$$out_{FLC} = \frac{\sum_i \alpha_i \cdot w_i}{\sum_i \alpha_i}, \qquad \text{(EQ 1-3)}$$

with α_i = weight of the i-th rule, w_i = consequent of the i-th rule, and i indexing all rules in the rule-base. This requires that the consequent w_i of each rule is a Fuzzy **singleton** (i.e. a Fuzzy set with only one value at which the membership function is 1.0).[1]

1.3 The SM^2 system

The Self Mobile Space Manipulator (SM^2) is a seven DOF, 1/3 scale version of a robot designed to work on the trusswork of Space Station Freedom. Scaling rules have been used to keep dynamic parameters of the scaled-down robot (such as natural frequencies, masses, stiffness) similar to those of the full-sized one. Designed for zero-gravity environment, SM^2 is very light-weight and very flexible. It provides mobility and manipulation capabilities and can assist astronauts in maintenance, inspection and transportation tasks.

1. Alternatively, w_i can be considered as the center of gravity of an arbitrary membership function. Neglecting overlap in the output membership functions, we use a center-of-gravity method in determining the output of the FLC.

As depicted in Figure 2, SM^2 is designed symmetrically, with two flexible links connected by a rotary joint, and a node and a part gripper on each side. For the work reported here, one node gripper is always attached to the trusswork, which makes the remaining chain of links to the opposite end-effector (part gripper) a seven DOF robot. This end-effector may be used for manipulating objects. Locomotion of SM^2 is performed by moving the unattached node gripper to a node, attaching it there, and releasing the previously attached node gripper. To simulate realistic working conditions, the robot is attached to a gravity compensation (GC) system, which supports the free end of the robot at two points. The GC works actively in the x-y - directions and passively in the z - direction.

SM^2 can be operated in two modes, *autonomous motion* and *teleoperation*. For developing Fuzzy friction compensation, we performed the experiments in the autonomous motion mode. The implemented Fuzzy Inverse Kinematic Mapping was tested in teleoperation mode. A diagram with relevant control modules for both operation modes is shown in Figure 3.

Figure 2 Model of SM^2 with enumeration of joints

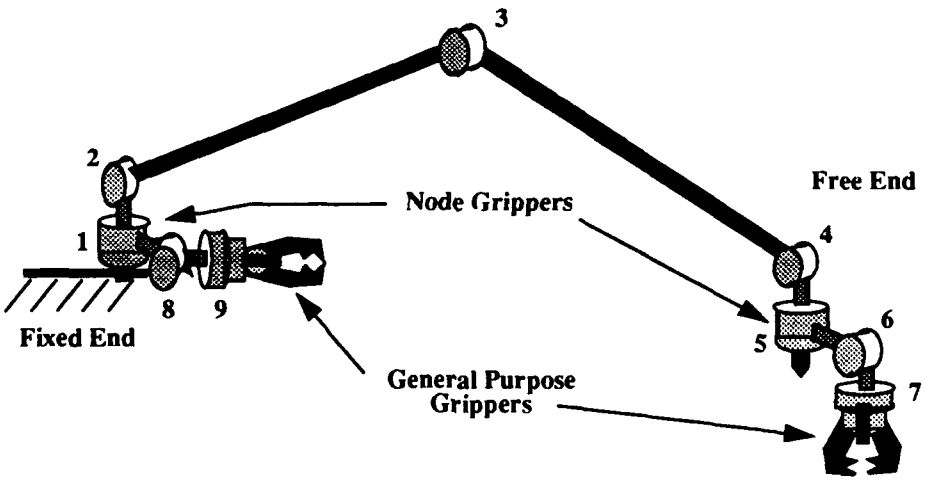
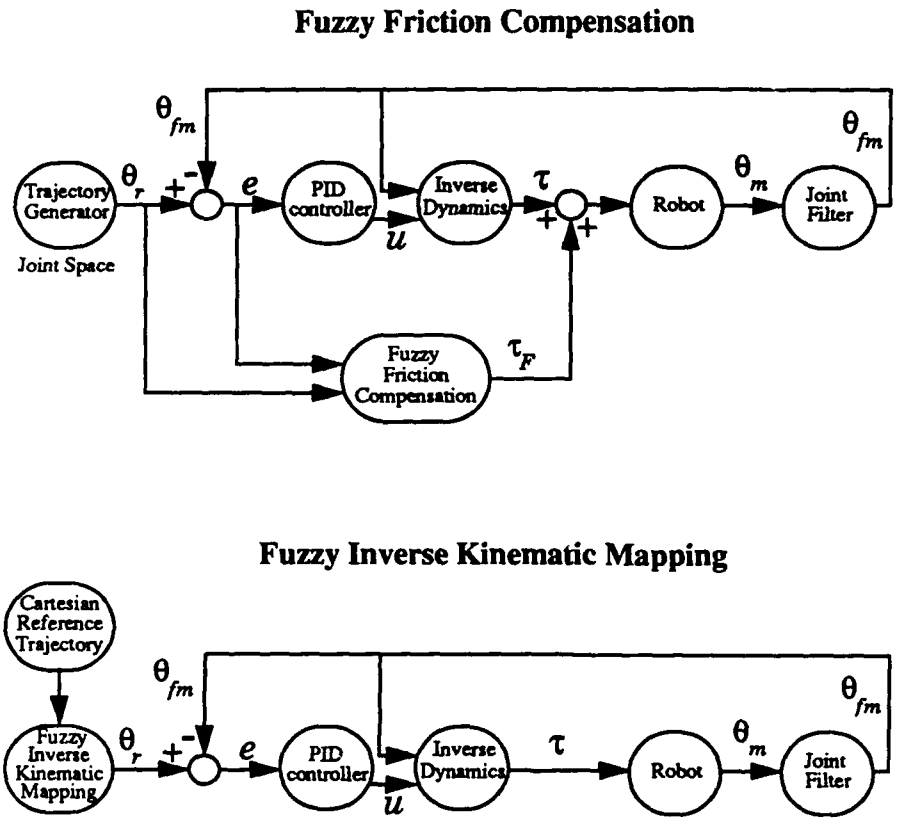


Figure 3 Relevant control modules in SM^2



2.0 Fuzzy Friction Compensation

In this chapter, we will briefly introduce friction theory and then propose fuzzy friction compensation schemes. Experimental results will be discussed, and the proposed schemes are compared to the scheme with constant friction compensation torque used in SM^2.

2.1 Friction theory

First, we will give a definition of Coulomb friction [12]:

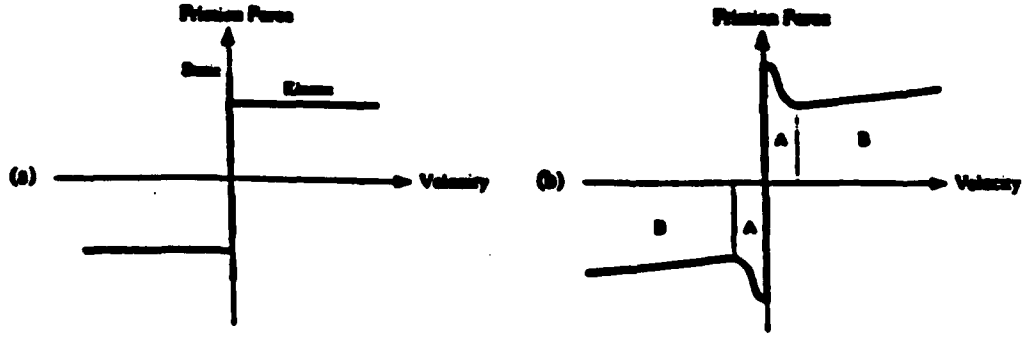
> Coulomb friction is a dissipative force that appears at the contact surface of two bodies in relative motion. The Coulomb friction force always opposes motion and its magnitude is proportional to the normal forces at the points of contact. The coefficient of proportionality is called the Coulomb friction coefficient.

We distinguish between static and dynamic friction. "Dynamic friction" refers to the case when there is relative motion, and "static friction" refers to the case when there is no relative motion.

> Dynamic Coulomb friction coefficient: this coefficient is denoted by μ and depends on the nature and the relative velocity of the bodies in contact.
>
> Static Coulomb friction coefficient: when there is no relative motion, the magnitude of the Coulomb friction force assumes the value required to ensure that relative motion will not occur. However, the value of the static Coulomb friction force cannot be greater than the normal force times the static Coulomb friction coefficient μ_s. [12]

Friction is difficult to model, since it is a highly non-linear function of several parameters, such as joint velocity, robot configuration, and payload. Due to the complexity of friction models, in robotics often an aggregate friction model is used. As an example, a simple model (the *kinetic/static model*) is shown in Figure 4, along with a more complex model (from [10]).

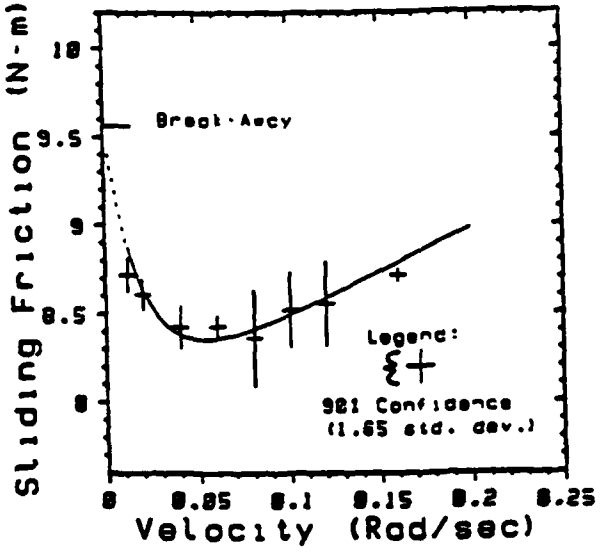
Figure 4 Friction force vs. velocity. (a) Kinetic / Static Model. (b) Complex Model



Armstrong [2] provided experimental evidence for the negative slope for very low velocities (area 'A' in Figure 4). The result is shown in Figure 5. This negative slope and a hys-

teresis effect ([10]) for very low velocities make stable control difficult. To describe the effect observed in Figure 5, Tustin's exponential model is usually used. Details can be found in [2] and [5].

Figure 5 Friction as a function of velocity at low velocities, fit with Tustin's exponential model



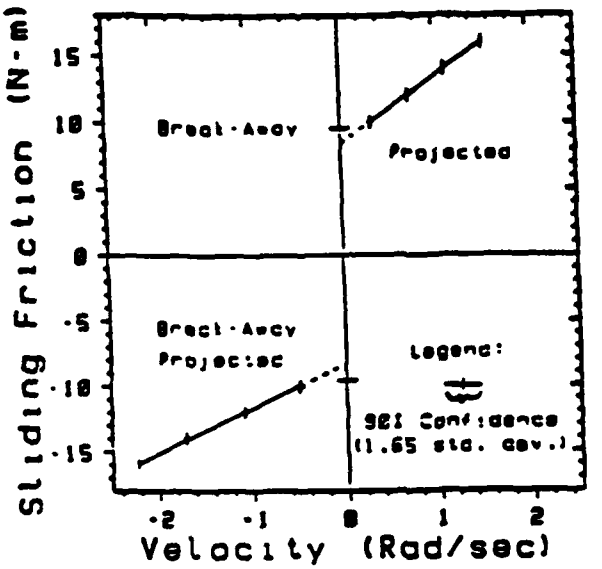
For moderate velocities, friction increases smoothly with the velocity. In this region (marked 'B' in Figure 4), friction is a combination of Coulomb friction and viscous friction [10]. Coulomb friction can be described by

$$F_{\text{friction}} = \mu |F_N| \text{sgn}(velocity) \qquad \text{(EQ 2-1)}$$

with F_N = normal force of contact and μ = coefficient of friction.

We see from (EQ 2-1) that friction forces are discontinuous at zero velocity [10]. Experimental validation for the *kinetic/static model* is shown in Figure 6 [2].

Figure 6 Friction torque as a function of velocity



Effect on control

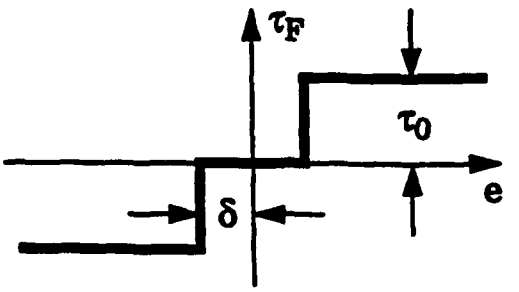
Friction manifests itself through various effects on control. If the control does not compensate for friction, we usually encounter large steady state errors and tracking lags. Due to additional nonlinear couplings among the joints, which are introduced by friction, and the complexity of the friction models, control is usually designed on basis of an inaccurate model.

Overcompensation in inexact models has been shown to lead to limit cycles [5]. Also, the discontinuity of the friction force near zero velocity may cause stick-slip-behavior. [10]

Friction compensation with constant torque

In SM^2, friction has been compensated by the scheme depicted in Figure 7. If the magnitude of the position error e exceeds a value δ, a constant friction compensation torque of value $\pm\tau_0$ is applied. The sign of the torque is equal to the sign of the input variable e.

Figure 7 Friction compensation with constant compensation torques



Motivation for Fuzzy Friction Compensation

The compensation scheme as shown in Figure 7 does not sufficiently represent the inherent nonlinearity of the relation of friction forces to configuration, payload and joint velocity. Though it reduces steady state error, it is not very efficient for dynamic errors. Furthermore, the empirical tuning of the parameters δ and τ_0 is a difficult task, since the quality of the compensation scheme varies significantly for small changes in the parameters.

2.2 Proposed schemes

We have designed friction compensation schemes based on some facts presented in the section on friction theory (see page 5). Different input parameters, such as position error, velocity error, velocity, and combinations of these, have been used in the proposed schemes. We designed a different set of membership functions for each joint, which accounts for different physical properties of the joints. The Fuzzy friction compensation schemes have been implemented in joints 1 to 3. These are the joints that contribute most to the position error.

Two schemes have been examined. The first one uses the FLC to generate an additional friction compensation torque which is added to the regular joint torque produced by the PD-controller and the inverse dynamics. We call this approach "Hybrid Fuzzy/PD control". The second method is based on a regular PD-controller structure, with the parameters tuned by the FLC. This approach is called "Fuzzy-tuning PD control".

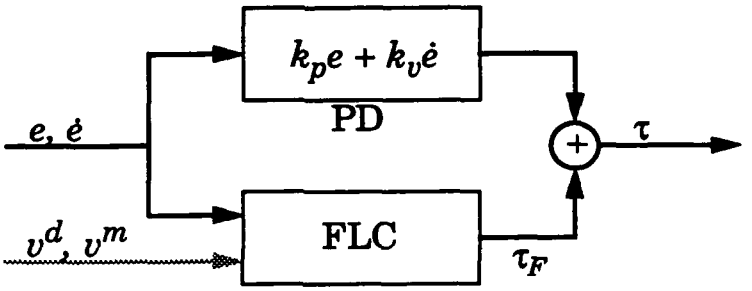
2.3 Hybrid Fuzzy/PD control

This method is represented in the diagram shown in Figure 8. Examined input parameters were the following entities and combinations thereof:

- position error e
- velocity error $\dot{e}$
- desired (reference) velocity v^d
- measured (actual) velocity v^m

The output of the FLC was added to the output torque of the PD controller. (PID controller was investigated, but led to weaker performance, especially as far as the steady state error is concerned.)

Figure 8 Structure of hybrid Fuzzy/PD control



(τ = applied torque, τ_p = friction compensation torque).

Considering that friction forces depend on the relative velocity of the two bodies in motion, the first question that arises here is: "Why do we use *position* error as input?"

Conceptually, a position error in our control structure with control cycles of constant length is equivalent to a relative velocity. The controller input is a position error e. Assuming ideal controller and ideal robot, the controller will issue a control command such that the error is zero in the next control cycle. The (ideal) robot will thus move the distance e within one control cycle T_c, which leads to a velocity v

$$v = e/T_c. \qquad \text{(EQ 2-2)}$$

Thus, the desired velocity of the robot is directly related to the position error.

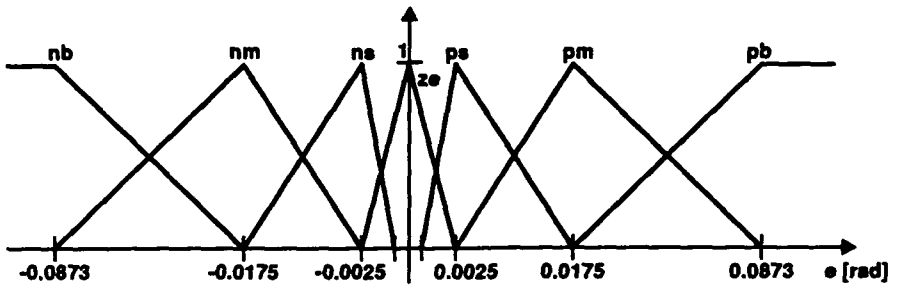
2.3.1 Position error as input

This scheme uses only position error as input to the FLC and will later be referred to as "hybrid error-only scheme". The position error was represented by the following fuzzy values:

NB	=	'negative big'
NM	=	'negative medium'
NS	=	'negative small'
ZE	=	'zero'
PS	=	'positive small'
PM	=	'positive medium'
PB	=	'positive big'

The membership functions for joint 1 can be found in Figure 9.

Figure 9 Membership functions for joint 1 - hybrid error-only scheme



The output variable was represented in the same categories (NM, ZE, PB,...). However, these fuzzy values for the output were defined as fuzzy singletons. The used values were normalized and evenly distributed between 0 and 1 (or between 0.5 and 1 for some experiments):

fuzzy value		output value
NB	:	-1.000
NM	:	-0.667
NS	:	-0.333
ZE	:	0.000
PS	:	0.333
PM	:	0.667
PB	:	1.000.

For joint 2 and 3, the membership functions were of the same structure, although with slightly different values.

All joints shared the same rule-base, which is presented in Table 1 .

Table 1 Hybrid error-only scheme: rule-base for joint 1 - 3

Error:	e_NB	e_NM	e_NS	e_ZE	e_PS	e_PM	e_PB
Output:	o_NB	o_NM	o_NS	o_ZE	o_PS	o_PM	o_PB

The main difference in the joints was the output scaling factor. To facilitate experiments and support a more unified representation, a normalized output had been derived in the FLC, which then was scaled by a scaling factor unique to each joint. Typical values were 4500, 2800 and 2800 for joint 1 to 3, respectively. We found that the steady state errors were reduced significantly, as compared to the scheme with constant friction compensation torque. This holds especially for joint 1.

It was a common experience in all experiments that the motion performance for joint 1 was improved the most by introducing Fuzzy friction compensation, and the performance was also more stable than for joint 2 and 3. This has been mostly attributed to the gravity compensation system. The GC is active in x-y - direction, which affects mainly joint 1. In

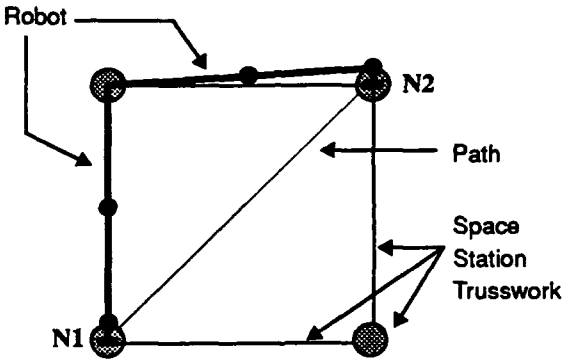
z - direction, the GC works passively (counterweight and pulleys). Also, the point of support is fixed. For these reasons, joint 2 and joint 3 are more affected by the deficiencies of the GC and hence show less improvement.

For the standard experiment, which consisted of moving from one node to another on a linear path (in Cartesian space) in five seconds, shown in Figure 10, we obtained the following averaged results:

Table 2 Hybrid error-only scheme: average results for standard experiment (5 sec)

Steady state error:	N1-->N2	N2-->N1	Average
Jt1	0.034	0.033	0.034
Jt2	0.076	0.102	0.089
Jt3	0.217	0.117	0.167

Figure 10 Hybrid error-only scheme: path for standard experiment



Further numerical and graphical results may be found in section 2.3.3 on page 12.

2.3.2 Position error and velocity as input

Now, we have considered a scheme that uses position error and reference/measured velocity as input parameters. The idea was to improve the dynamic performance of the system by reducing tracking lags. Velocity can represent one of the robot states, and this information can help to adjust the compensation scheme. For example, the compensation torques might be increased for very low velocity to overcome static friction. To consider the new input parameters, the rule-bases had to be adjusted.

The use of the measured velocity has not proven to be very promising. Although conceptually appealing, we could not use the current robot state as given in the position and velocity values to predict the motion tendency and accordingly modify our output. The use of measured velocity led very easily to unstable behavior of the robot. For stable behavior, a trade-off between both considerable overshoot and steady state error had to be made.

Using the reference velocity instead of the measured one improved stability significantly. On the other hand, the necessity for a trade-off between acceptable overshoot and acceptable steady state error was still present, although to a slightly smaller extent. It should be noted, however, that, as expected, the dynamic errors, as compared to the scheme with only position error as input, were reduced significantly for both measured and reference velocity as input. To give an idea of the changes, some figures will follow. Comparing the scheme with reference velocity and position error to the scheme with only position error, the

dynamic error[1]	decreased by	30%
maximum position error	decreased by	40%
steady state error	increased by	735%

Though the steady state error rose relatively much, the new result with 0.285° is still well within our limit of 0.5°.

Position error and velocity error as input variables

Using both position error and velocity error as input parameters to the FLC didn't yield any promising results at all. This scheme was not investigated any further.

2.3.3 Comparison study

To evaluate the performance of the Fuzzy friction compensator, a comparison study was conducted for three types of friction compensation schemes:

- PD-controller with constant friction compensation torque ('PD')
- PID-controller with constant friction compensation torque ('PID')
- PD-controller with fuzzy friction compensation ('FFC')

The controllers were individually tuned to yield highest performance.

The experiment we conducted was the movement of the robot tip from one node to another, on a linear path, with constant tip orientation (see Figure 10). The major movement was carried out by joint 1. Due to the linear path and the constant tip orientation, the trajectory also involved changes in joint 2 and joint 3 position.

In the following Table 3 , we see the performance for each compensation type. The results correspond to the movement from node N1 to node N2, from N2 to N1, and the average of these two results. The experiment was carried out in five seconds. Steady state position error was measured at fifteen seconds after the start of the experiment.

1. as given by the integral over the position error magnitude

Table 3 Comparison of different compensation schemes - Steady state position error [in degrees] for a movement from one node to another in 5 seconds

Controller type	Joint 1			Joint 2			Joint 3		
PD	7.283	4.574	**5.929**	0.515	0.077	**0.296**	0.887	0.332	**0.609**
PID	0.120	1.339	**0.730**	0.046	0.539	**0.293**	0.314	0.051	**0.183**
Fuzzy Logic	0.034	0.033	**0.034**	0.076	0.102	**0.089**	0.217	0.117	**0.167**

Table 4 contains the results of the same experiment, now performed in ten seconds. The robot moves here at very low speed, and therefore the effect of static friction is significant. Furthermore, the membership functions had been developed for a five second movement. Steady state error was again measured at fifteen seconds after the start of the experiment.

Table 4 Comparison of different compensation schemes - Steady state position error [in degrees] for a movement from one node to another in 10 seconds

Controller type	Joint 1			Joint 2			Joint 3		
PD	7.632	5.268	**6.446**	1.784	1.615	**1.699**	3.345	3.021	**3.183**
PID	0.496	1.886	**1.191**	0.501	0.603	**0.552**	3.264	2.290	**2.777**
Fuzzy Logic	0.014	0.060	**0.037**	0.289	0.800	**0.545**	1.348	0.485	**0.917**

To help visualize the improvement in the results, we present the averaged results again as pair-wise comparison of the different compensation schemes. The values denote the ratios of the second result with respect to the first one.

Table 5 Relative error for different compensation schemes in 5 sec - experiment

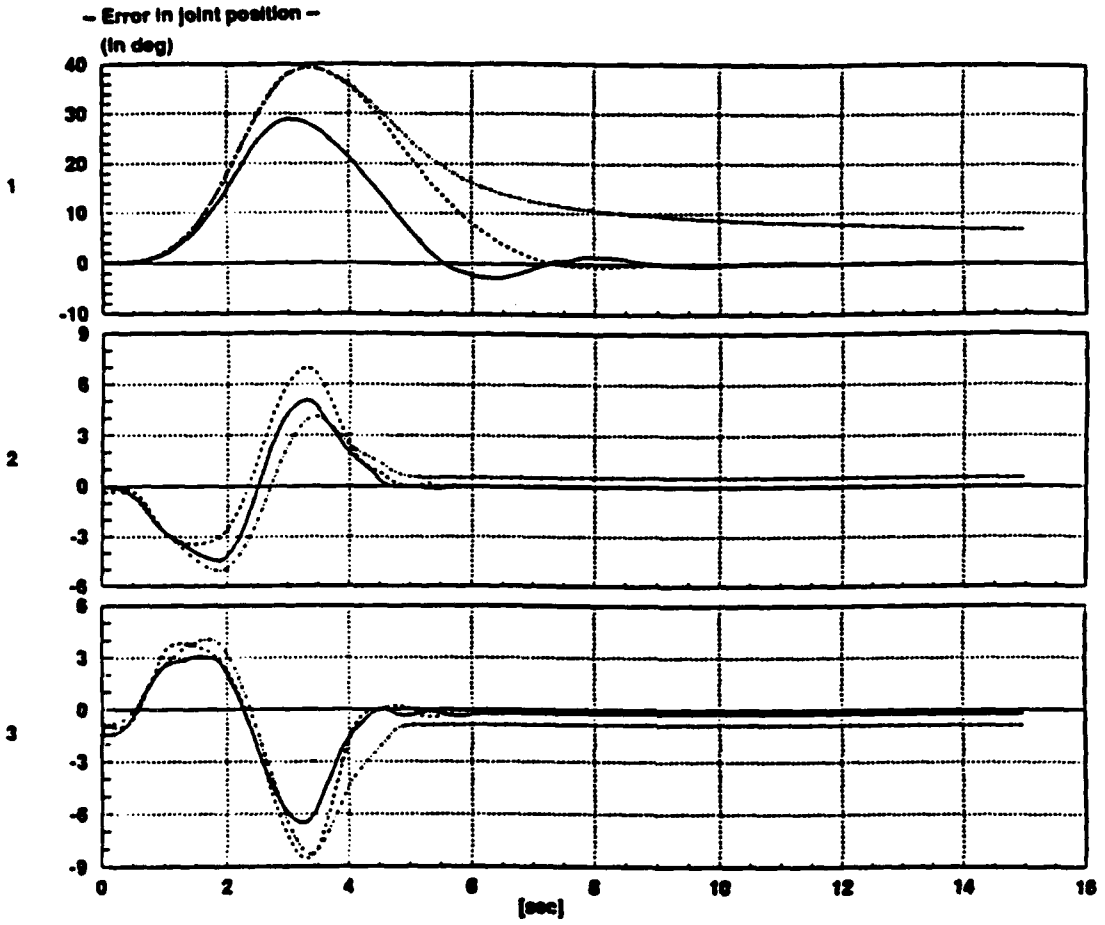
Compared schemes	Joint 1	Joint 2	Joint 3
PID/PD	12.31 %	98.99 %	30.05 %
FFC/PD	0.57 %	30.07 %	27.42 %
FFC/PID	4.66 %	30.37 %	91.26 %

Table 6 Relative error for different compensation schemes in 10 sec - experiment

Compared schemes	Joint 1	Joint 2	Joint 3
PID/PD	18.48 %	32.49 %	87.24 %
FFC/PD	0.57 %	32.08 %	28.81 %
FFC/PID	3.11 %	98.73 %	33.02 %

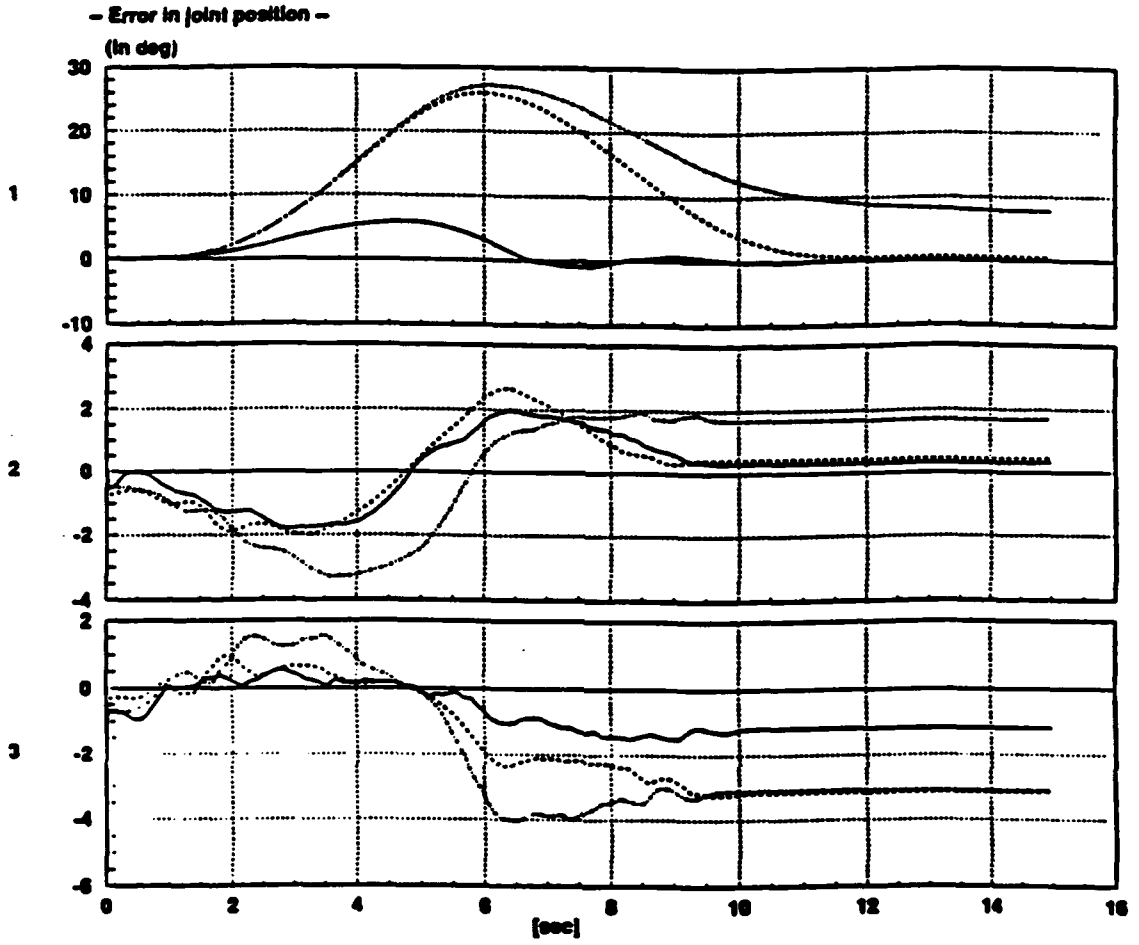
The comparison shall be concluded by some plots that correspond to the numerical results.

Figure 11 Comparison of steady state position error for different friction compensation schemes - 5 sec - Experiment



The solid line denotes FFC, the dashed line PID, and the dotted line PD.

Figure 12 Comparison of steady state position error for different friction compensation schemes - 10 sec - Experiment



The solid line denotes FFC, the dashed line PID, and the dotted line PD.

2.3.4 Summary

For joint 1, a significant reduction in steady state position error (to 0.6 %) could be achieved when compared to the PD controller with constant friction compensation torque. The steady state position error for joint 2 and joint 3 was reduced to approximately 30%. Compared to PID control with constant friction compensation torque, the result showed a reduction to values between 30% and 90% of the original value. The results differ between the two movements (N1->N2, N2->N1) due to residual gravity effects.

2.4 Fuzzy-tuning PD control

The second approach uses an FLC to tune the parameters k_p and k_v of a conventional PD-controller:

$$u = k_p e + k_v \dot{e} \qquad \text{(EQ 2-3)}$$

with u = control signal (which is used to compute the torques τ), e = joint position error, and k_p and k_v = PD parameters, tuned by the FLC.

We realized soon that the scheme wouldn't reach the performance of the hybrid error-only scheme, unless we included some further information.

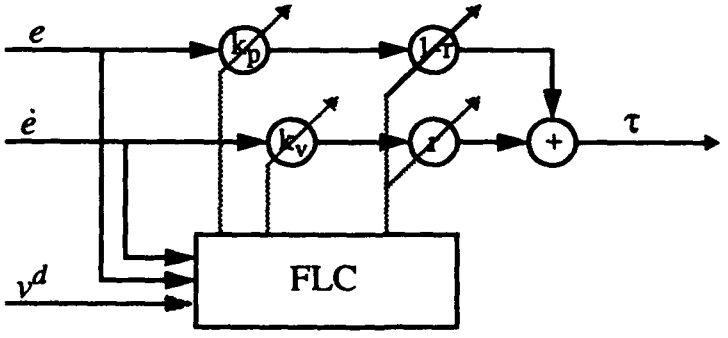
Since we try to reduce both steady state and dynamic errors, we set priorities as follows:

- at the beginning and especially at the end of the trajectory, the position error is more important.
- in the middle of the trajectory, we are more concerned about the velocity error.

This additional "knowledge" was incorporated in a parameter that we called 'rate' r. The rate was used to weigh the $k_p e$ and $k_v \dot{e}$ contributions, and it was also tuned by the FLC. The present scheme is now as shown in Figure 13. The output of the scheme is

$$\tau = (1 - r) k_p e + r k_v \dot{e} \qquad \text{(EQ 2-4)}$$

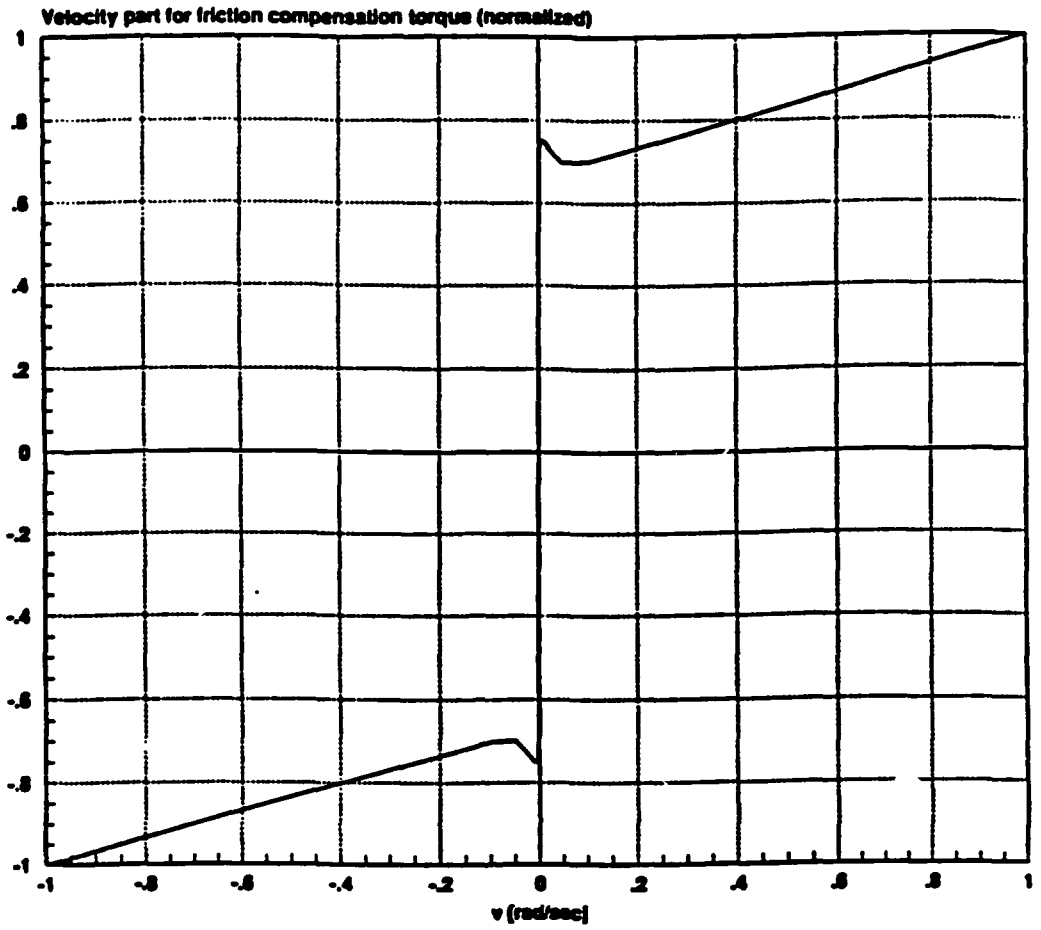
Figure 13 Fuzzy-tuning PD controller



The parameters have been modeled as follows:

- k_p was constant for large values and increased for small velocities. This ensured that $k_p e$ was above a minimum value, until e dropped under a very low value.
- k_v was designed such that $k_v \dot{e}$ matched the dynamic friction model. Actually to make the design easier, we replaced the term $k_v \dot{e}$ by $k_v \cdot (2 \cdot \mathrm{sgn}(\dot{e}) + \dot{e})$ which can be rewritten as $k_v \cdot \mathrm{sgn}(\dot{e}) \cdot (2 + |\dot{e}|)$. This allowed us to easily model torques like the one shown in Figure 14.

Figure 14 Velocity error contribution to the friction compensation torque



Compared to the hybrid error-only scheme, we achieved an improvement in dynamic error (as represented by the integral over the error magnitude), but the result for the steady state error was not as good. Compared to the hybrid error-only scheme[2], the

dynamic error	decreased by	21%
maximum error	decreased by	23%
steady state error	increased by	410%.

The value for the steady state error was 0.17°, which is still well within the objective of 0.5°, and well below the result of the PID controller with constant friction compensation torque (see Table 3 (p. 13)). When compared to the hybrid error-velocity scheme presented in section 2.3.2 on page 11, we see that the performance of this scheme cannot be reached as far as the dynamic error is concerned. However, for the steady state error, this PD-tuning scheme is better. Furthermore, the PD-tuning scheme reduces dynamic error by 55% when compared to a PID controller with constant friction compensation torque.

2. These values apply to joint 1.

Two other points are in favor of the PD-tuning scheme when compared to the hybrid error-velocity scheme. First, the results for the steady state error were much more consistent for this method (i.e. the variance was much smaller). Secondly, this scheme was very stable, in the sense that it was quite insensitive to the values of the output-scaling factors. The hybrid error-velocity scheme on the other hand was very sensitive. For that scheme, it was not easy to find output-scaling factors that would show an acceptable performance and though not easily lead to oscillations. The method presented here didn't display this effect. It was much easier to find parameters for good performance.

2.5 Summary

We have developed two Fuzzy friction compensation schemes. The hybrid Fuzzy-PD scheme with position error only as input yields the best results for steady state error. If dynamics errors are of concern, and the steady state error is less important, the Fuzzy-tuning PD scheme is desirable. Compared to PID controller with constant friction compensation, both proposed schemes achieve much better results in steady state error, and they also improve the dynamic behavior.

3.0 Fuzzy Inverse Kinematic Mapping

In this chapter, we will, after a brief review of Inverse Kinematics, propose a Fuzzy Inverse Kinematic Mapping. We will analyze the scheme and present simulation results. Finally, the implementation of the Fuzzy Inverse Kinematic Mapping for teleoperation of SM^2 is discussed.

3.1 Introduction

3.1.1 Inverse Kinematics

A robot task is usually specified in Cartesian coordinates for the end-effector of the robot, while the control of the robot is usually performed in robot joint space. To bring these two worlds together, mappings from one space to the other are necessary. The transformation from joint space to Cartesian space is called *forward kinematics*, and that from Cartesian space to joint space is called *Inverse Kinematics*.

Given the geometric model of the robot, i.e. link lengths, joint angles, and a description how the links are joined together, the *forward kinematic mapping* is determined by

$$x = f(\theta), \tag{EQ 3-1}$$

where x is a vector containing the Cartesian position and orientation of the robot, θ is a vector with the joint angles, and f represents the geometric model.

The inverse kinematic mapping,

$$\theta = f^{-1}(x) \tag{EQ 3-2}$$

however, is more difficult, since (EQ 3-1) is highly nonlinear, and that equation might not be invertible for certain configurations (singular positions), or may have multiple solutions.

Example

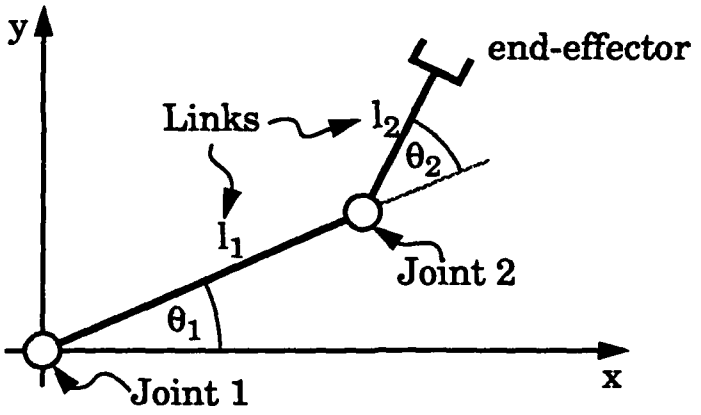
The Inverse Kinematics is illustrated by a simple planar, non-redundant example.

Referring to Figure 15, the *forward kinematics* can be written as

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) \\ l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) \end{bmatrix} \tag{EQ 3-3}$$

where l_i denote the link lengths.

Figure 15 Robot with two links in a planar world



Given the coordinates of the end-effector position, the joint angles can be derived as follows (*Inverse Kinematics*):

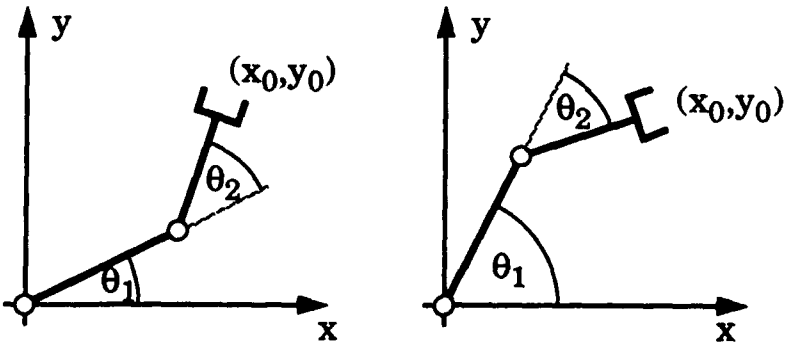
$$\theta_2 = \text{acos}\frac{x^2+y^2-l_1^2-l_2^2}{2l_1l_2} \qquad \text{(EQ 3-4)}$$

$$\theta_e = \text{asin}\frac{l_2\sin\theta_2}{\sqrt{x^2+y^2}}$$

$$\theta_1 = \text{atan2}(y,x) - \theta_e$$

Equations (EQ 3-4) do not yield a unique solution. If (θ_1, θ_2) is a solution, $(\theta_1 + 2\theta_e, -\theta_2)$ is also a configuration with the same end-effector position. These two solutions correspond to the "left-handed" and the "right-handed" configurations as illustrated in the following Figure 16.

Figure 16 Left-handed and right-handed configuration of a 2-link-robot.



In the configuration on the left ("right-handed"), θ_2 is positive. The "left-handed" configuration on the right refers to $\theta_2 < 0$.

3.1.2 Kinematic redundancy

Redundant robot refers to a mechanism that provides more degrees of freedom than the task requires. The additional degrees of freedom are used for obstacle avoidance, keeping joints within their physical limitations, placing the joint torques closest to the midpoints of the joint torque limits, optimization of performance, dexterous manipulation, singularity avoidance and other reasons.

For a redundant robot, we differentiate equation (EQ 3-1) with respect to time:

$$\dot{x} = J(\theta)\,\dot{\theta} \qquad \text{(EQ 3-5)}$$

where $\dot{x} \in R^m$, $\dot{\theta} \in R^n$ and the Jacobian $(J = \frac{df}{d\theta}) \in R^{m \times n}$. The number or degree of redundancy is thus $n - m$. For our time-discrete controller, equation (EQ 3-5) can be rewritten as

$$\Delta x = J(\theta)\,\Delta\theta \qquad \text{(EQ 3-6)}$$

One class of methods for the resolution of redundancy in Inverse Kinematics uses a pseudo-inverse[1] approach. The solution usually has the form

$$\dot{\theta} = J^{+}\dot{x} + (I - J^{+}J)\,z \qquad \text{(EQ 3-7)}$$

where $J^{+} = J^{T}(JJ^{T})^{-1}$ and z is an arbitrary vector. The first term of (EQ 3-7) represents the solution of minimum norm joint velocity vector, the second term is a projection of z onto the null space of J and can be used for optimization purposes.

The main shortcoming for this class of methods is the fact that the solution can in general only be obtained numerically, which is computationally expensive. Hence, these methods are oftentimes not suitable for real time applications. Furthermore, these methods operate at the velocity, and not at the position level. Global properties of the workspace cannot easily be derived.

Another class of schemes is based on optimization of some motion criteria, like minimal joint motion, or minimum energy. These schemes, like the pseudo-inverse based methods, are numerical, and have the same disadvantageous properties.

3.2 Fuzzy Inverse Kinematic Mapping (FIKM)

3.2.1 Basic concept

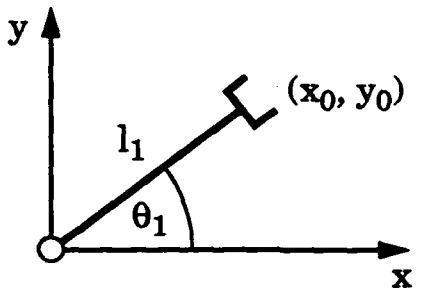
The Fuzzy Logic approach to the inverse kinematic problem works with a simple rule-base. The ideology of this rule-base can be illustrated by the following example.

1. also called Moore-Penrose generalized inverse.

Considering a planar robot with *one link*, as shown in Figure 17, the end-effector can be described as

$$x_0 = l_1 \cos\theta_1$$
$$y_0 = l_1 \sin\theta_1 \qquad \text{(EQ 3-8)}$$

Figure 17 Simple one-link planar robot.



Linearization around a given joint angle θ_1 yields the following forward kinematic equations:

$$dx = c_x d\theta_1 \quad \text{or} \quad \dot{x} = c_x \dot{\theta}_1$$
$$dy = c_y d\theta_1 \qquad \dot{y} = c_y \dot{\theta}_1 \qquad \text{(EQ 3-9)}$$

with

$$c_x = -l_1 \sin\theta_1$$
$$c_y = l_1 \cos\theta_1 \qquad \text{(EQ 3-10)}$$

As can be seen, the values of c_i are the elements of the Jacobian matrix $J = \frac{df}{d\theta}$ with $(x, y)^T = f(\theta_1)$.

Using the Fuzzy values

pm = "positive medium"
nl = "negative large"
ns = "negative small",

a sample rule for Inverse Kinematics might be formulated as

IF *dx* is *pm* **AND** c_x is *nl* **THEN** $d\theta_1$ is *ns*,

which is intuitively clear: when the link is pointing more or less in the direction of the positive *y*-axis, which is equivalent to saying c_x is nl, and we try to move the end-effector a small amount in the direction of the positive *x*-axis (*dx* is pm), we should decrease θ_1 a little ($d\theta_1$ is ns).

This concept can now be explored for different combinations of dx and c_x, and the rules may be represented in the form of a table as shown in Table 7 . Since the same argumentation holds for dy and c_y the same table can be used, replacing *dx* by *dy* and c_x by c_y

Table 7 Sample rulebase for simple Fuzzy Inverse Kinematic Mapping

c_x \ dx	NB	NS	ZE	PS	PB
NM	PB	PS	ZE	NS	NB
NS	PVB	PM	ZE	NM	NVB
ZE	ZE	ZE	ZE	ZE	ZE
PS	NVB	NM	ZE	PM	PVB
PM	NB	NS	ZE	PS	PB

The Fuzzy values are to be interpreted as follows:

ZE = 'zero"
P... = "positive ..."
N... = "negative ..."
...S = "... small"
...M = "... medium"
...B = "... big"
...V... = "... very ..."

For example, the Fuzzy value PVB should be read as "positive very big".

The presented rule-base has been derived for a one-link planar robot. However, if a *n*-link serial robot is considered as composed of *n* links similar to the presented one, the above rule-base applies to each of these links.

Furthermore, since the rule-base has been derived using a linearized model, the principle of superposition can be applied and is used to combine the contributions of the rule-base of each joint for the given *dx* and *dy*. This is the basis for our implementation of the Fuzzy Inverse Kinematic Mapping.

3.2.2 Realization

The rule-base we actually used for the IKM was defined over slightly different Fuzzy variables. The coefficient c_{ij} is now the element of the Jacobian matrix in the *i*-th row and *j*-th column. In other words, it relates the *i*-th component of the vector *dx* with the *j*-th joint angle θ_j. The c_{ij}-s are defined over the Fuzzy values

PM PS ZE NS NM

with the symbols defined as above (Table 7).

These Fuzzy values are shown in Figure 18. The values for the elements of the Cartesian displacement vector dx are defined over the Fuzzy values

PB PM PS PVS ZE NVS NS NM NB

These values are shown in Figure 19.

Figure 18 Membership functions for the elements of the Jacobian matrix, c_{ij}.

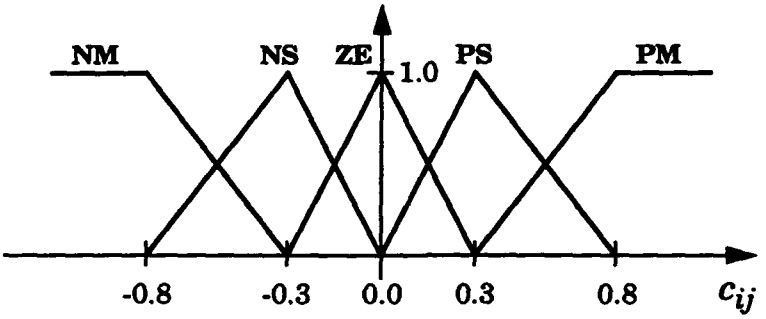
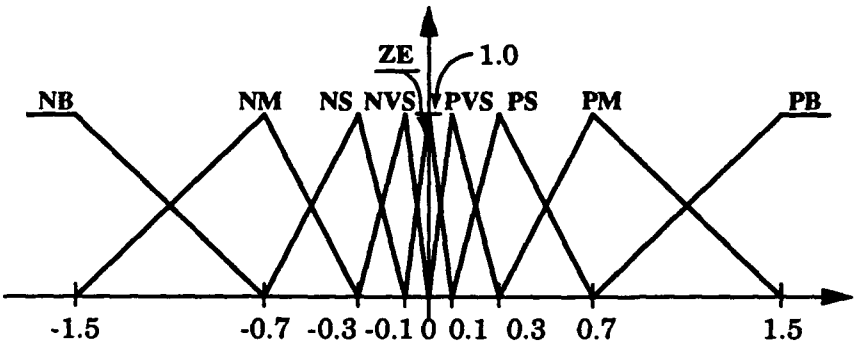


Figure 19 Membership functions for the elements of the displacement vector dx



The rule-base we actually used is given in the following Table 8 .

Two remarks about this rule-base are in place. First, all values for $d\theta_j$ have been determined such that the centroid value for $d\theta_j$ is approximately the ratio of the centroid values of dx_i and c_{ij}. Actually, for all possible combinations of dx_i and c_{ij}, the ratio of their centroid values has been determined, and similar ratios have been aggregated to one (average) value, in order to keep the representation clearer.

Secondly, for $c_{ij} = 0$, the $d\theta_j$ may be arbitrary, since $c_{ij}d\theta_j = 0$, which means it doesn't affect the displacement in dx_i-direction. The effective $d\theta_j$ is a combination of the $d\theta_j$

determined for the different dx_i, and thus there will usually be some $c_{ij} \neq 0$, which will yield some result for $d\theta_j$.

Table 8 Implemented rulebase for Fuzzy Inverse Kinematic Mapping

		dx_i								
	$d\theta_j$	NB	NM	NS	NVS	ZE	PVS	PS	PM	PB
c_{ij}	NM	PB	PM	PS	PVS	ZE	NVS	NS	NM	NB
	NS	PVB	PB	PM	PS	ZE	NS	NM	NB	NVB
	ZE					ZE				
	PS	NVB	NB	NM	NS	ZE	PS	PM	PB	PVB
	PM	NB	NM	NS	NVS	ZE	PVS	PS	PM	PB

Determination of $d\theta_j$

Let us now discuss in detail how the obtained $d\theta_j$ are combined. We recall that for each combination of dx_i and c_{ij} we determine a $d\theta_j$. Let us denote this result of the Fuzzy mapping by $d\theta_{ij}^+$:

$$d\theta_{ij}^+ = \mathrm{FM}(c_{ij}, dx_i), \qquad \text{(EQ 3-11)}$$

where FM stands for the Fuzzy mapping using the membership functions defined in Figure 18 and Figure 19 and the rule-base of Table 8 .

Furthermore, let us define **weights** as

$$w_{ij} = |c_{ij}|. \qquad \text{(EQ 3-12)}$$

For scaling, we need the column- and the row-sums of these weights, which we will denote by

$$\mathrm{wjsum}_i = \sum_{j=1}^{n} w_{ij}, \qquad \text{(EQ 3-13)}$$

the sum in row i over all joints (this gives the 'j'), and similarly

$$\mathrm{wcsum}_j = \sum_{i=1}^{m} w_{ij}, \qquad \text{(EQ 3-14)}$$

which is the sum in column j over all Cartesian coordinates.

Since each $d\theta_{ij}^{+}$ yields approximately dx_i, we are scaling the contributions of each $d\theta_{ij}^{+}$ to dx_i by some factor, such that the sum of the contributions is close to dx_i:

$$\sum_{j=1}^{n} a_{ij} d\theta_{ij}^{+} \approx dx_i, \qquad \text{(EQ 3-15)}$$

in other words,

$$\sum_{j=1}^{n} a_{ij} = 1 \qquad \text{for all } i = 1 \ldots m \qquad \text{(EQ 3-16)}$$

The effective joint angle change is accordingly

$$d\hat{\theta}_{ij} = a_{ij} d\theta_{ij}^{+} \qquad \text{(EQ 3-17)}$$

Now we consider one particular joint j. Fo dx_i, we have derived some $d\theta_{ij}^{+}$. We average these values by again using some weigh.. b_{ij} such that the sum of all b_{ij} is one:

$$\sum_{i=1}^{m} b_{ij} = 1 \qquad \text{for all } j = 1 \ldots n \qquad \text{(EQ 3-18)}$$

The effective joint angle change is then

$$d\theta_j = \sum_{i=1}^{m} b_{ij} d\hat{\theta}_{ij} \qquad \text{(EQ 3-19)}$$

The weights b_{ij} are chosen under the following assumption: starting from a particular $d\theta_j$ and considering $d\theta_j + \delta$, the change in dx_i is highest for the dx_i where the $|c_{ij}|$ $(i = 1 \ldots m)$ is the largest. To keep errors small, the averaged $d\theta_j$ should be closest to those $d\hat{\theta}_{ij}$ whose corresponding $|c_{ij}|$ is large. Therefore, a weighted average scheme is appropriate, with $w_{ij} = |c_{ij}|$ being the weight.

Hence,

$$b_{ij} = w_{ij} / \text{wcsum}_j. \qquad \text{(EQ 3-20)}$$

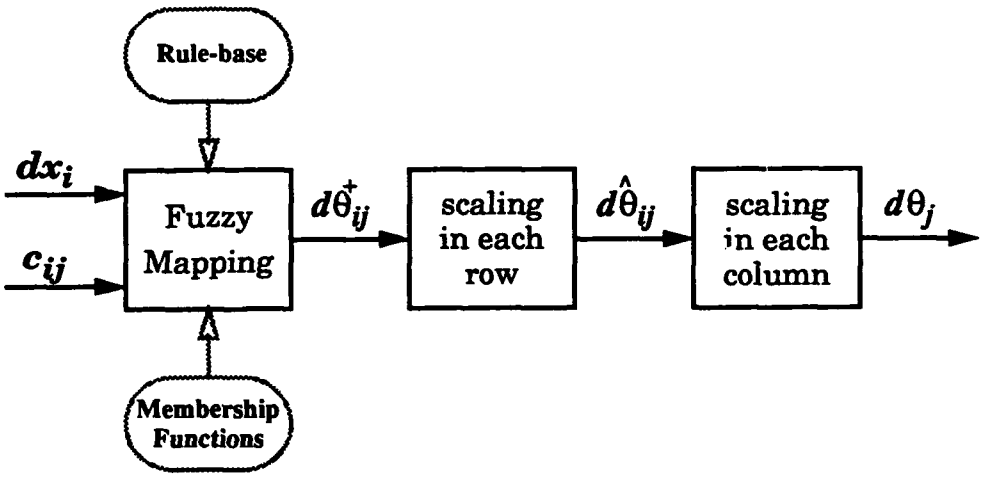
A similar argumentation yields the determination of the a_{ij} as

$$a_{ij} = w_{ij} / \text{wjsum}_i \qquad \text{(EQ 3-21)}$$

To summarize the weighing scheme, it can be noted that the results of the Fuzzy mappings are scaled both in each column and each row, with the respective scaling factors being the weight w_{ij} divided by the sum of the weights in this column / row.

The scheme, as developed hitherto, is summarized in the following Figure 20.

Figure 20 The basic Fuzzy Inverse Kinematic Mapping with scaling.



3.2.3 Discussion

In this section, we will show why the scheme works, and where the limitations of the scheme can be found.

We recall that $d\theta^{+}_{ij}$, the results of the Fuzzy mapping, are mapped to $d\theta_j$ by

$$d\theta_j = \sum_{i=1}^{m} \frac{w_{ij}}{\mathrm{wcsum}_j} \frac{w_{ij}}{\mathrm{wjsum}_i} d\theta^{+}_{ij}. \qquad \text{(EQ 3-22)}$$

Considering how the rule-base is generated, we can approximate

$$d\theta^{+}_{ij} \approx \frac{dx_i}{c_{ij}}. \qquad \text{(EQ 3-23)}$$

In fact, for the centroids of the membership functions, the maximum error in this equation is approximately 15% (which arises from the aggregation of several Fuzzy values for $d\theta^{+}_{ij}$ to one Fuzzy value). Since we can only show that the scheme is working qualitatively, we use the approximation in (EQ 3-23) as exact equation.

Inserting (EQ 3-23) into (EQ 3-22) and considering

$$\frac{|a|^2}{a} = \frac{a\,\mathrm{sgn}\,(a)\,|a|}{\mathrm{sgn}\,(a)\,|a|} = a, \qquad \text{(EQ 3-24)}$$

we arrive at

$$d\theta_j = \sum_{i=1}^{m} \frac{c_{ij}}{\text{wcsum}_j \text{wjsum}_i} dx_i. \qquad \text{(EQ 3-25)}$$

Lets see how these $d\theta_j$ amount to $\overline{dx_i}$ (we are using dx_i now for the desired dx_i, which is the input to the Fuzzy Inverse Kinematic Mapping, and $\overline{dx_i}$, which is the result of the mapping, the actual commanded Cartesian displacement.)

$$\overline{dx_i} = \sum_{j=1}^{n} c_{ij} d\theta_j = \sum_{j=1}^{n} \sum_{i=1}^{m} \frac{c_{ij}}{\text{wjsum}_i} \frac{c_{ij}}{\text{wcsum}_j} dx_i \qquad \text{(EQ 3-26)}$$

For more insight, we explore an example with $n = 2$ and $m = 3$. Then, (EQ 3-26) results in

$$\overline{dx_1} = \mu_{11} dx_1 + \mu_{12} dx_2 \qquad \text{(EQ 3-27)}$$

where

$$\mu_{11} = \left(\frac{|c_{11}|}{1 + \frac{|c_{21}|}{|c_{11}|}} + \frac{|c_{12}|}{1 + \frac{|c_{22}|}{|c_{12}|}} + \frac{|c_{13}|}{1 + \frac{|c_{23}|}{|c_{13}|}} \right) \frac{1}{|c_{11}| + |c_{12}| + |c_{13}|} \qquad \text{(EQ 3-28)}$$

and

$$\mu_{12} = \left(\frac{c_{21}\,\text{sgn}(c_{11})}{1 + \frac{|c_{21}|}{|c_{11}|}} + \frac{c_{22}\,\text{sgn}(c_{12})}{1 + \frac{|c_{22}|}{|c_{12}|}} + \frac{c_{23}\,\text{sgn}(c_{13})}{1 + \frac{|c_{23}|}{|c_{13}|}} \right) \frac{1}{|c_{21}| + |c_{22}| + |c_{23}|} \qquad \text{(EQ 3-29)}$$

Ideally, we would like to have

$$\mu_{11} = 1 \quad \textbf{and} \quad \mu_{12} = 0 \qquad \text{(EQ 3-30)}$$

As we see from (EQ 3-28), $\mu_{11} \approx 1$ for $|c_{2j}| \approx 0$ and $|c_{1j}| > 0$, or for $|c_{1j}| \gg |c_{2j}|$. The range for μ_{11}, however, is the interval $[0;1]$. What we can derive is that at least the contribution of dx_1 to $\overline{dx_1}$ has the correct sign. The contribution of dx_2 to $\overline{dx_1}$, i.e. $\mu_{12} dx_2$, can be positive or negative. Considering the whole workspace, the average contribution is zero.

To conclude, we see that the scheme works well, although there are configurations in which the performance of the scheme is suboptimal.

3.3 Simulation study

3.3.1 Simulation system

We have developed a simulation system that allows Fuzzy Inverse Kinematic Mapping to be carried out for an n-link planar robot.

The main **features** of the program are

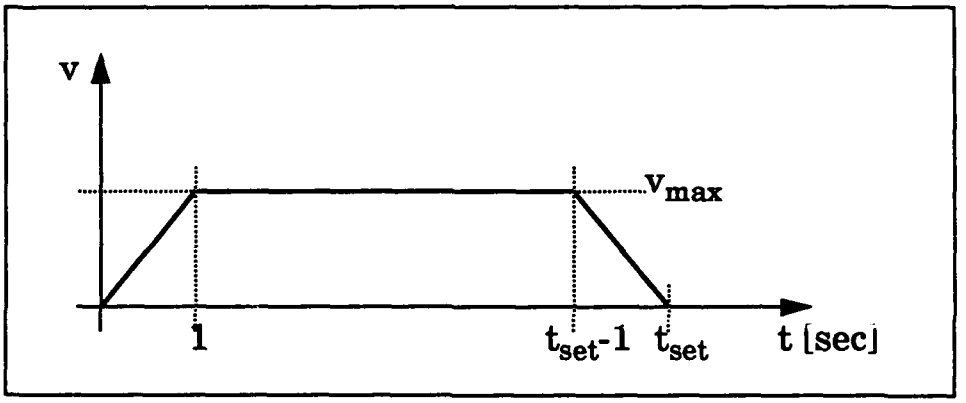
- to generate a reference trajectory
- to simulate FIKM
- to display and save results
- to adjust data display and data recording

The following **parameters** could be chosen interactively:

- link lengths
- trajectory type
- interpolation type
- initial and final position, and, as required, an intermediate position
- 'control' type

One type of **trajectories** could be a straight line between two points, with linear interpolation and either constant velocity or with a velocity profile as depicted in Figure 21.

Figure 21 Velocity profile for trajectory with linear interpolation

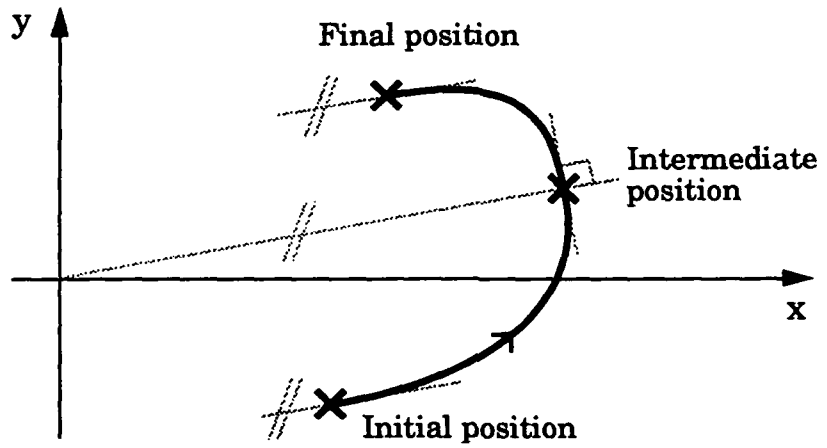


Velocity increases linearly during the first second, and decreases linearly during the last second. In between, the velocity is constant. At t_{set}, the robot is desired to have reached the target position.

In Figure 21, the **settling time** t_{set} is the time at which the robot should ideally reach the target position.

Another type of trajectory is a **3-point trajectory**, with an initial position, an intermediate position and a target position. The interpolation is done in Cartesian space, and the trajectory is composed of two quarts of an ellipse. An example is given in Figure 22. The intermediate position is passed on a radial path with respect to the origin, i.e. the velocity vector at the intermediate position is perpendicular to the vector from the origin to the intermediate position. This was introduced for testing the Inverse Kinematic Mapping when the desired trajectory goes through a singularity, in particular the one which is encountered when the robot is fully stretched out (for a multi-link robot).

Figure 22 Sample 3-point trajectory



The **interpolation** for the linear trajectories can be performed in Cartesian space or in joint space.

According to the interpolation type (joint or Cartesian space), the **final position** was also given in joint space or in Cartesian space, respectively. The **initial position** has always been defined in joint space.

The simulation can be run in two modes. In the first mode, a number of steps can be given, and the path is then divided into this number of steps, which gives the desired dx_i-s. The second mode assumes a **controller** which controls a robot such that the commanded dx_i and the actual (performed) dx_i were the same (i.e. no dynamics). The controller frequency f_c and the desired settling time t_{set} can be chosen. As in our SM^2 application, the controller is servoing to the target position even after t_{set}. At the time t_{end}, the simulation was stopped. Typical values were derived from SM^2 autonomous motion control:

$$f_c = 40 \text{ Hz}, \quad t_{set} = 5 \text{ sec}, \quad t_{end} = 7 \text{ sec}$$

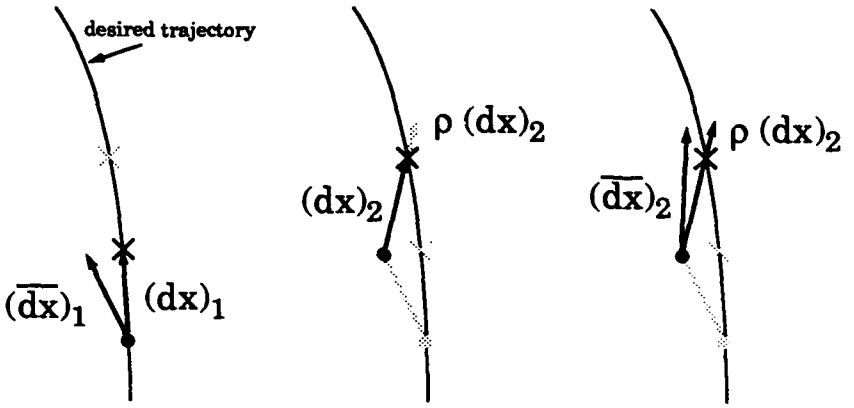
3.3.2 Modifications of the scheme

The scheme as presented works well. However, it can be improved by introducing three additional processing steps:

- amplification of the commanded dx depending on the error in the previous step
- oscillation detection and suppression
- lowpass filtering of the desired dx

The ideas was the following: if our scheme works fine, we shouldn't change anything. However, if the error starts growing, we should increase the tendency to move in the correct direction by amplifying the desired dx. For example, if the scheme tends to command a position on the left side of the desired path, the next dx will have a component which is pointing to the right, towards the path. By amplifying this dx by some factor ρ, we obtain a desired position on the right of the path. Given that the scheme still tends to be too far on the left, as before, we will be very close to the path. This is illustrated in Figure 23.

Figure 23 Effect of amplification of dx depending on error



How the amplification of dx based on error works: Left: desired $(dx)_1$ and commanded $(\overline{dx})_1$; Middle: resulting desired $(dx)_2$; Right: amplified $(dx)_2$ and commanded $(\overline{dx})_2$, resulting in a position closer to the desired trajectory.

If the error grows too large, however, we have to assume that the scheme is doing very wrong and we should move very cautiously. Therefore, for large errors, the amplification factor is reduced to small values (less than 1.0).

There are occasions when the scheme commands a larger dx than desired. If this situation persists, then the correcting action might also be too large, resulting in an even larger error, and so forth. This might lead to oscillations. Due to our amplification factor, which goes down when the error becomes large, this effect is usually dampened out.

On the other hand, to be sure that this attenuation is effective and is quickly in effect, we try to detect oscillations. If an oscillation is detected, the amplification factor is reduced significantly.

We consider only one type of oscillation, which is when the commanded $d\theta_j$ keeps changing its sign. The implementation is as follows. For all joints, the signs of the $d\theta_j$ are compared to the previous ones. If one of the signs of the $d\theta_j$ has changed, a flag is set. We use

a ring buffer with ten entries where this flag is set or reset. If four or more flags are set (which means that among the recent ten control cycles, there have been four or more cycles with changes in some signs of the $d\theta_j$) this is interpreted as the detection of an upcoming oscillation.

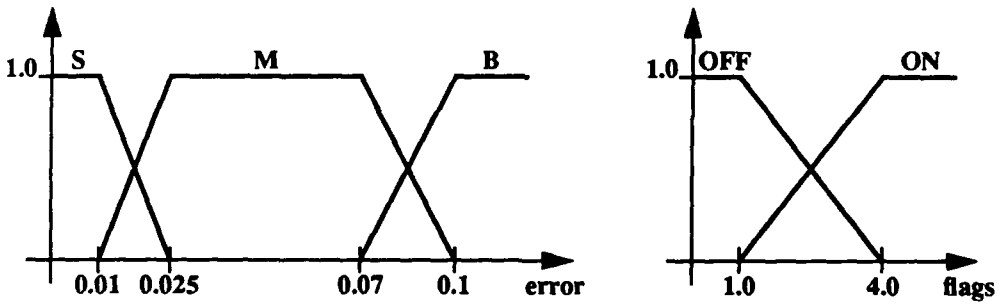
We cannot detect oscillations of the form:

$$d\theta_j = \ldots + 0.2, \quad + 0.4, \quad + 0.2, \quad + 0.4, \ldots \qquad \text{(EQ 3-31)}$$

As for the threshold of *four* flags, we have seen that the choice is not very critical. Any threshold between 3 and 5 is reasonable and yields virtually the same results. There are only very rare situations where the number of flags climbs to four (or three, or five), and stays there for a while. Usually, the number of flags is zero or one, and when oscillations start up, the number goes straight to 10. A value above five is not recommended, since the main consequence is that it takes longer to detect an oscillation.

Both the error in the last move and the number of flags were the input to a Fuzzy mapping which produced the amplification factor ρ. The corresponding membership functions are given in Figure 24. The rule-base can be found in Table 9 .

Figure 24 Determination of amplification factor ρ: Membership functions for input variables



Membership functions for the input variables error (on the left), which is defined as the distance between the desired and the actual location, and for the oscillation (on the right), which is measured by the number of flags that are set in the ring buffer (see text for details on the flags).

Table 9 Rulebase for the determination of the amplification factor ρ

		Error:		
	ρ	S	M	B
Oscillations:	OFF	ONE	B	M
	ON	VS	S	VVS

The output singletons are defined as follows:

ONE	=	1.0
B	=	2.6
M	=	0.8
VS	=	0.1
S	=	0.4
VVS	=	0.03

Note: To smoothen the effect of the amplification factor ρ, especially when oscillations are being detected, we use a lowpass-filtered ρ:

$$\rho_{k,\,applied} = 0.33\rho_{k,\,FuzzyOutput} + 0.67\rho_{k-1,\,applied} \qquad \text{(EQ 3-32)}$$

Finally, the third type of additional processing we perform is a lowpass filtering of the desired *dx*:

$$(dx)_{k,\,applied} = c\,(dx)_{k,\,desired} + (1-c)\,(dx)_{k-1,\,applied} \qquad \text{(EQ 3-33)}$$

A typical value for c was 0.4 .

3.3.3 Simulation Results

Comparison study for different modifications of the scheme

To show the effect of oscillation detection and *dx*-input filtering on the tracking accuracy, we compare the maximum tracking error and the integral of the error magnitude.

The schemes to be compared are:

1. scheme with amplification of *dx* depending on the error in the previous step
2. same as 1), with additional oscillation detection and suppression
3. same as 2), with additional *dx*-input lowpass filtering ($c = 0.40$).

All trajectories were of type 'linear with velocity profile' and were run with the controller parameters:

f	=	40 Hz
t_{set}	=	5 sec
t_{end}	=	7 sec

The last trajectory was also examined for $f = 60$ Hz and 80 Hz.

If oscillation occurred, the result for maximum error is marked with a 'star' *. Very light oscillation is marked with a 'plus' +. Very light oscillation only during the last two seconds (servoing to target position, desired velocity is zero) is marked with a 'minus' -.

Table 10 Maximum error for sample trajectories for different modifications of the proposed scheme

Scheme \ Trajectory	1	2	3	4	5	6	6/60Hz	6/80Hz
1	4.3539*	0.5318⁻	5.2557*	0.8723⁻	11.1331*	7.8246*	9.5996*	8.1395*
2	1.8271⁺	0.6200⁻	2.1070⁺	0.8723	29.7351*	13.4819*	30.8794*	24.7492*
3	1.1286	0.7577	0.6762⁺	0.9173	3.2417⁻	1.5679⁻	1.1606⁻	0.9609⁻

Table 11 Integral error for sample trajectories for different modifications of the proposed scheme

Scheme \ Trajectory	1	2	3	4	5	6	6/60Hz	6/80Hz
1	9.9007	1.9169	18.7532	2.2479	26.9389	26.8828	26.5717	26.6042
2	4.0329	1.9614	2.2995	2.2479	17.8769	14.0296	14.3730	12.9900
3	2.2858	1.9512	2.0092	2.2707	6.2828	5.2561	3.8923	3.3162

As can be seen, scheme 3) improves the results considerably for most trajectories, especially for those that had a large error. For trajectories with small error (#2 and #4), the results are not as good as for those with large error.

The increase in the maximum error by using oscillation detection without lowpass filtering of *dx* can be attributed to the way the oscillation suppression works. Since the $\Delta\theta_j$ are scaled down significantly when oscillations are coming up, the robot movement also is slowed down considerably, and the tracking lag and thus the error increase.

A large error does not necessarily mean that the robot is not following the reference path. Since the error is defined as the difference between reference and actual position, a time lag (robot is too slow, but on the reference path) contributes equally to the error as a deviation from the path.

It should be noted that although oscillation detection increased the maximum error, the integral error decreased, for most trajectories significantly (except for a minimal increase for trajectory #2).

The lowpass filtering of *dx* decreased **both** maximum error **and** integral error for most trajectories.

Graphical results

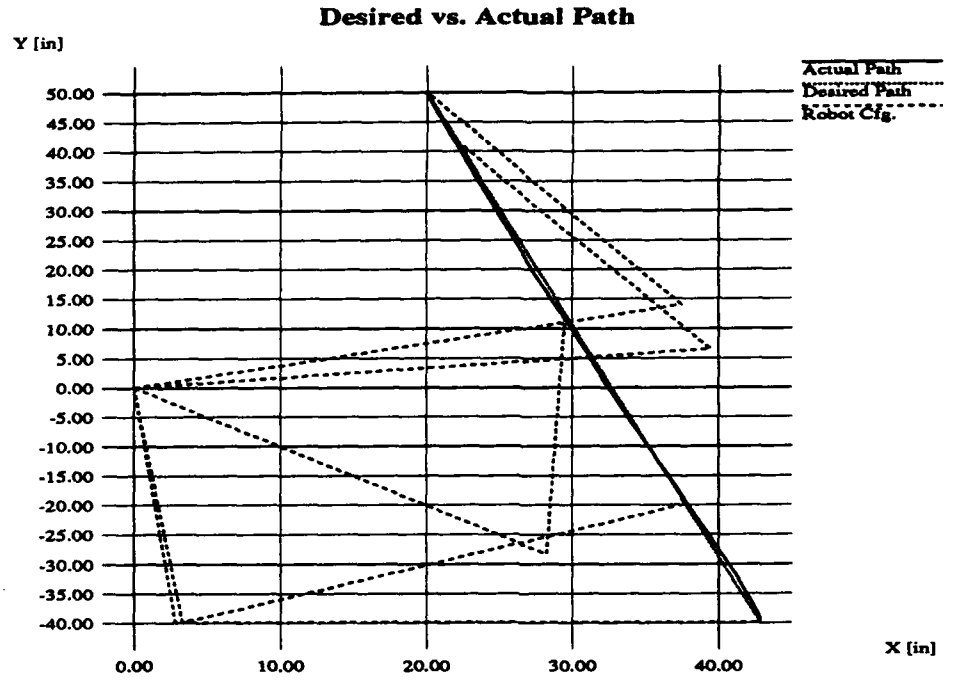
In the following pages, plots of simulation results are presented. The simulations were conducted with the following parameters:

- two-link robot:
  $l_1 = 40$ in
  $l_2 = 40$ in
- three-link robot (uses actual SM^2 dimensions):
  $l_1 = l_2 = 38.23$ in
  $l_3 = 4.55$ in
- both: controller with
  $f = 40$ Hz
  $t_{set} = 5$ sec
  $t_{end} = 7$ sec

The trajectories were either linear with velocity profile or 3-point elliptic. Interpolation has always been executed in Cartesian space.

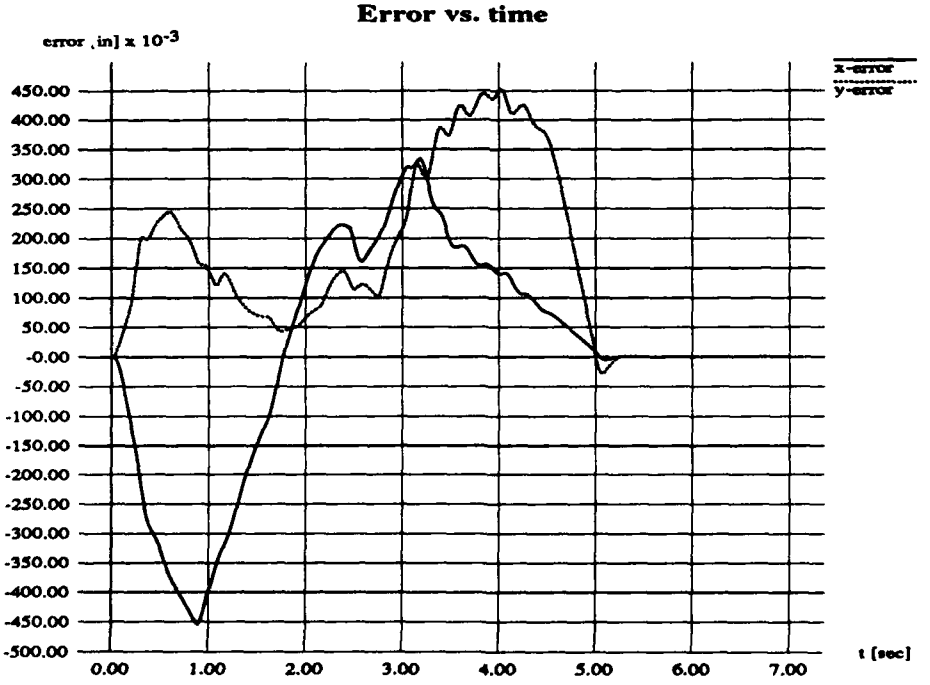
Shown are the actual vs. the reference path, the error in both dimensions (x, y), and the numerical results.

Figure 25 2-link robot - Example 1: the path



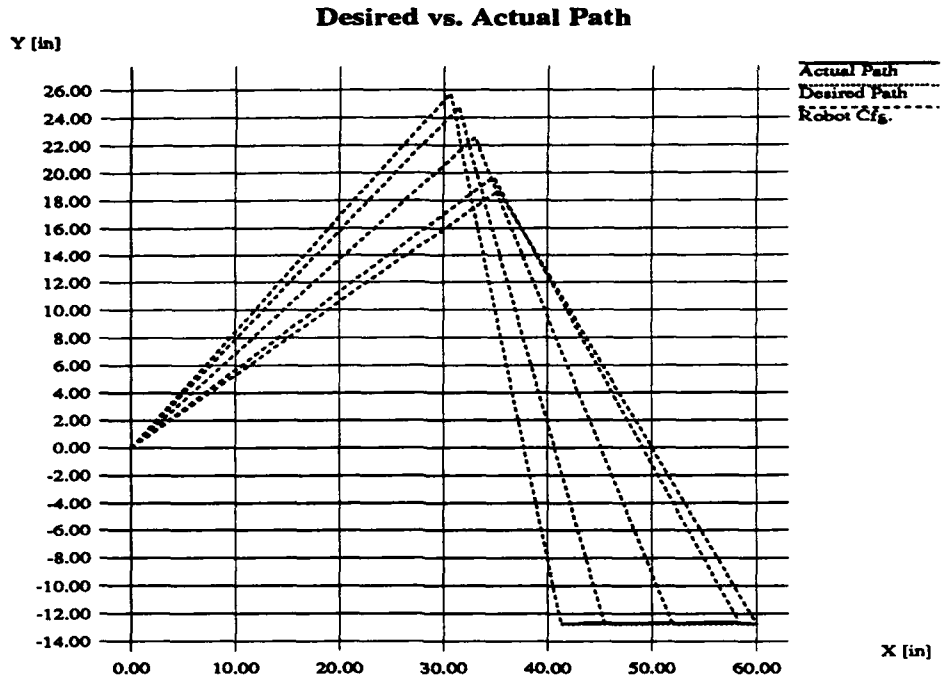
Initial position: $\theta_0 = [-1.5,1.5]^T$; Final position: $x_{end} = [20.0,50.0]^T$.

Figure 26 2-link robot - Example 1: the error



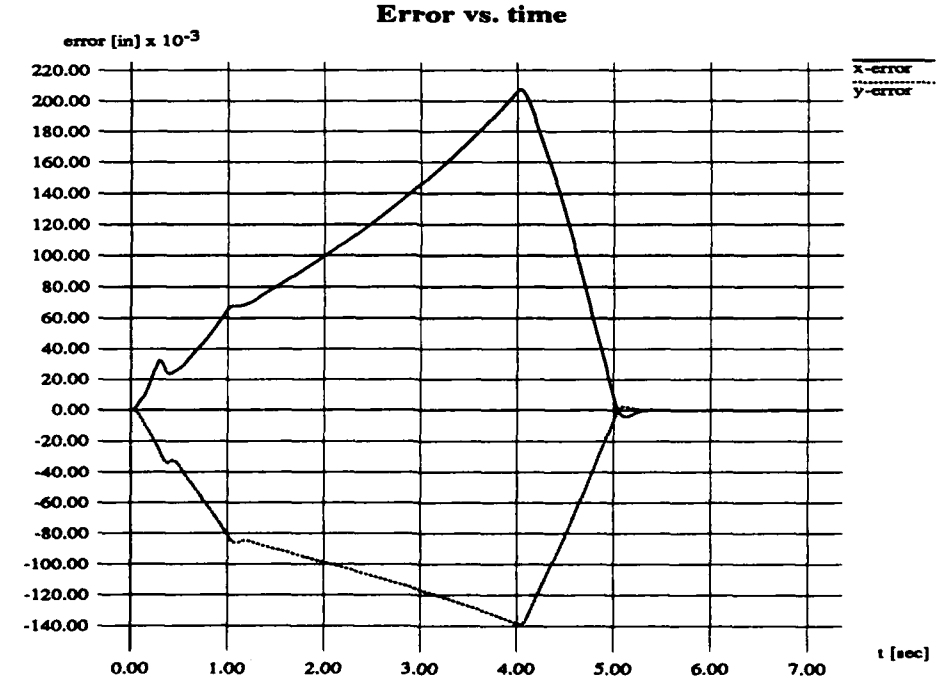
Maximum error:	[0.4547, 0.4531]	|.| = 0.4817
Error integral:	[0.9646, 1.0564]	|.| = 1.5309
Settling time error:	[0.0064, -0.0073]	|.| = 0.0097
Final error:	[0.0000, vt0.0000]	|.| = 0.0000

Figure 27 2-link robot - Example 2: the path



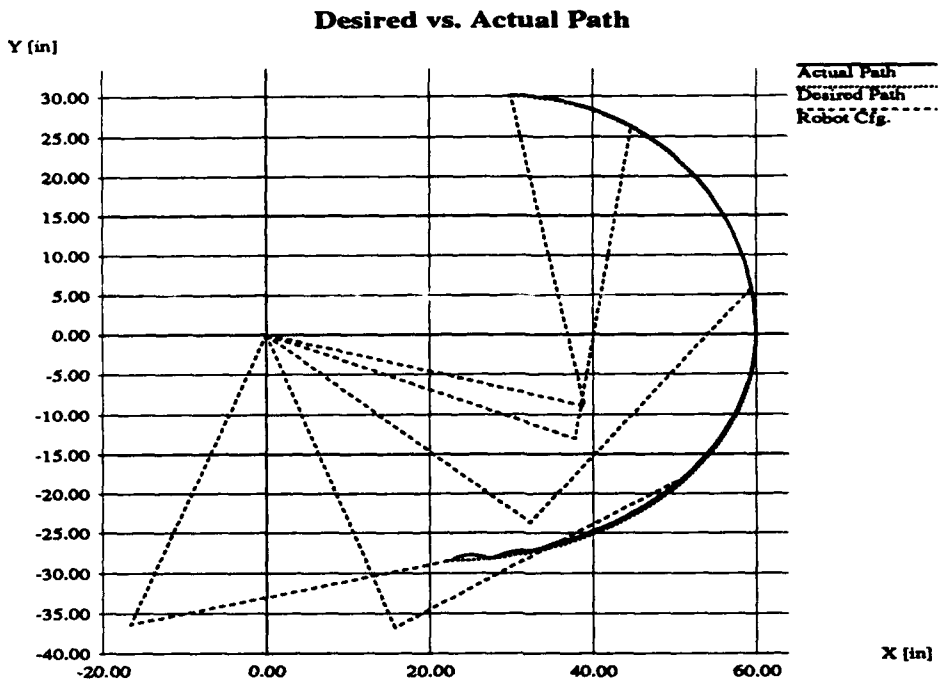
Initial position: $\theta_0 = [0.7, -2.0]^T$; Final position: $x_{end} = [60.0, -12.774]^T$.

Figure 28 2-link robot - Example 2: the error



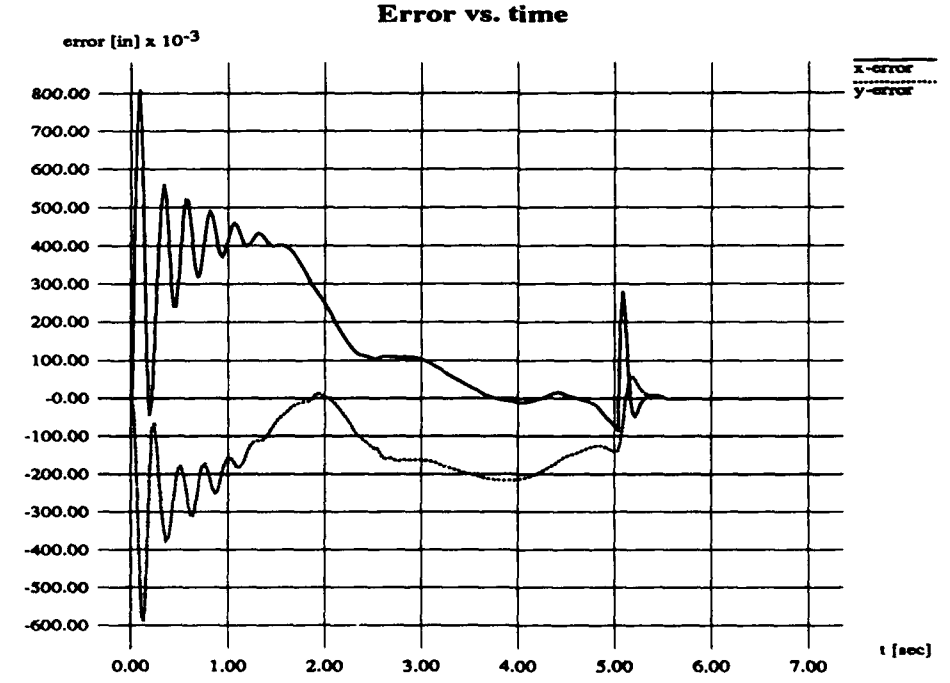
Maximum error:	[0.2077, 0.1390]	|.| = 0.2499
Error integral:	[0.5311, 0.4437]	|.| = 0.6962
Settling time error:	[0.0063, -0.0047]	|.| = 0.0078
Final error:	[0.0000, 0.0000]	|.| = 0.0000

Figure 29 2-link robot - Example 3: the path



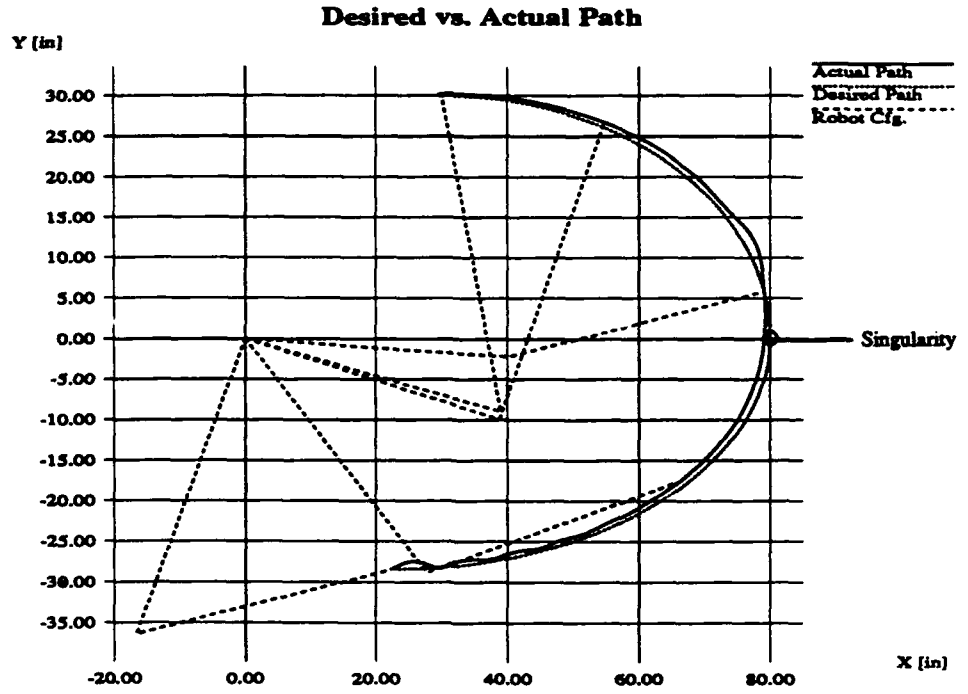
Initial position: $\theta_0 = [-2.0, 2.2]^T$; Intermed.pos. $x_{int} = [60.0, 0.0]^T$, Final position: $x_{end} = [30.0, 30.0]^T$.

Figure 30 2-link robot - Example 3: the error



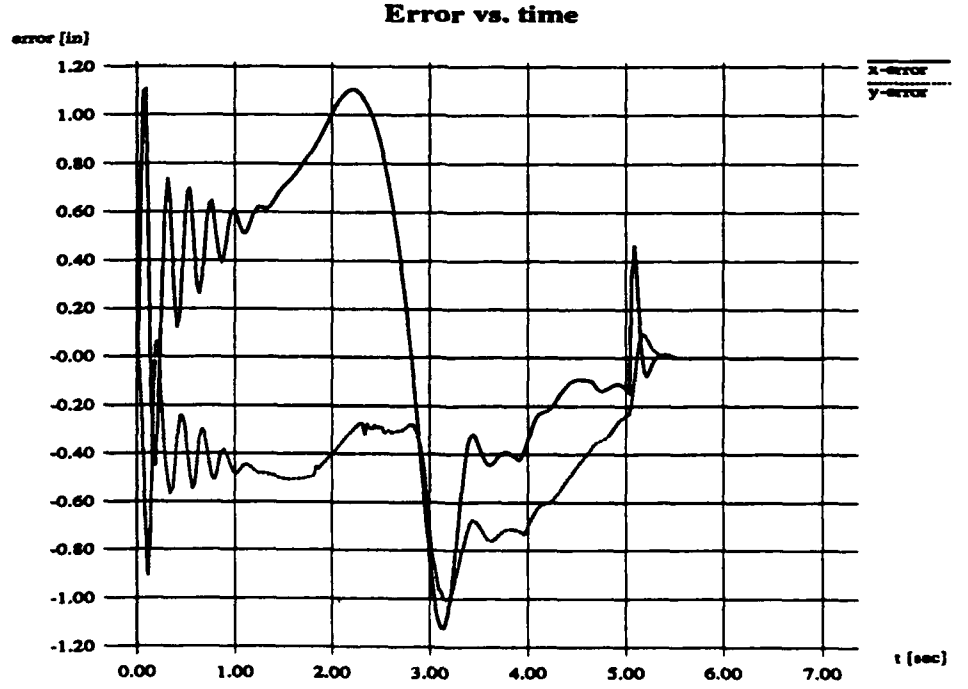
Maximum error:	[0.8104, 0.5914]	$\|.\| = 0.9178$
Error integral:	[0.9856, 0.8109]	$\|.\| = 1.4541$
Settling time error:	[-0.0769, -0.1400]	$\|.\| = 0.1597$
Final error:	[0.0000, 0.0000]	$\|.\| = 0.0000$

Figure 31 2-link robot - Example 4: the path



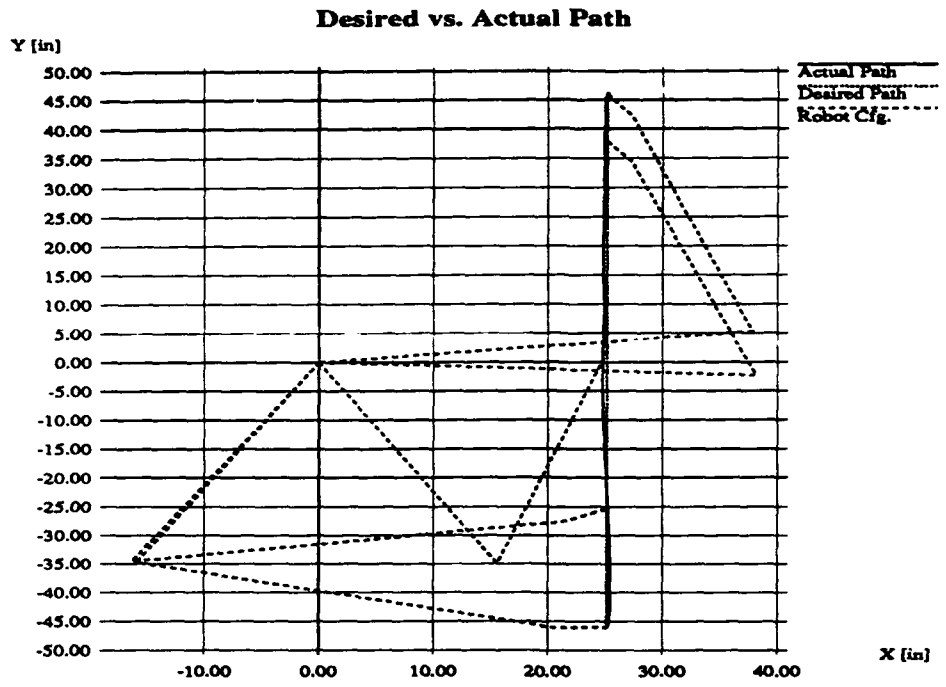
Initial position: $\theta_0 = [-2.0, 2.2]^T$; Intermed.pos. $x_{int} = [80.0, 0.0]^T$, Final position: $x_{end} = [30.0, 30.0]^T$.

Figure 32 2-link robot - Example 4: the error



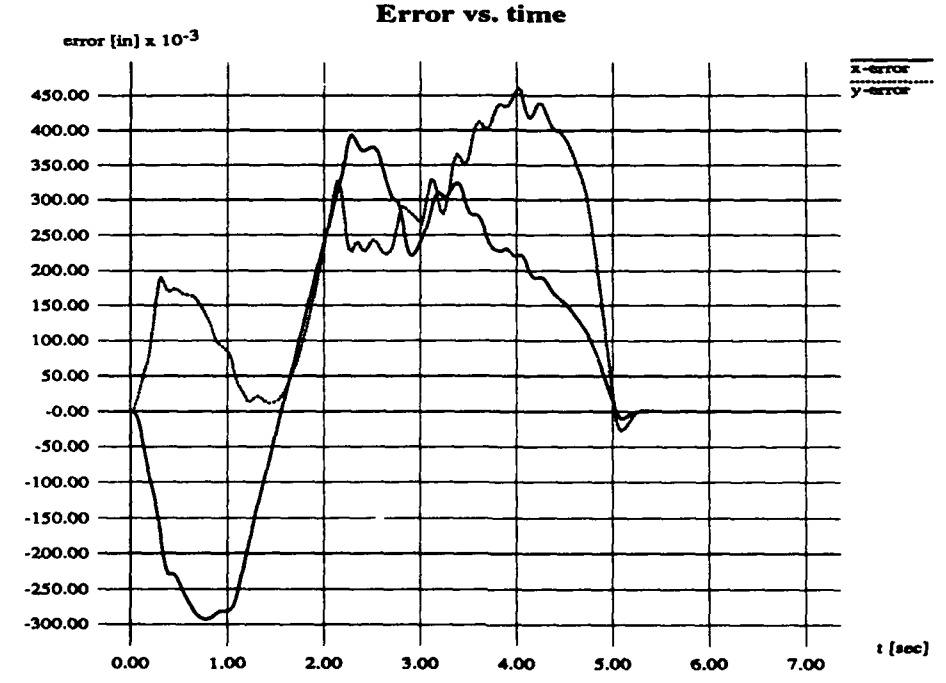
Maximum error:	[1.1217, 1.0040]	|.| = 1.4974
Error integral:	[2.7337, 2.4791]	|.| = 3.8909
Settling time error:	[-0.1310, -0.2398]	|.| = 0.2733
Final error:	[0.0000, 0.0000]	|.| = 0.0000

Figure 33 3-link robot - Example 1: the path



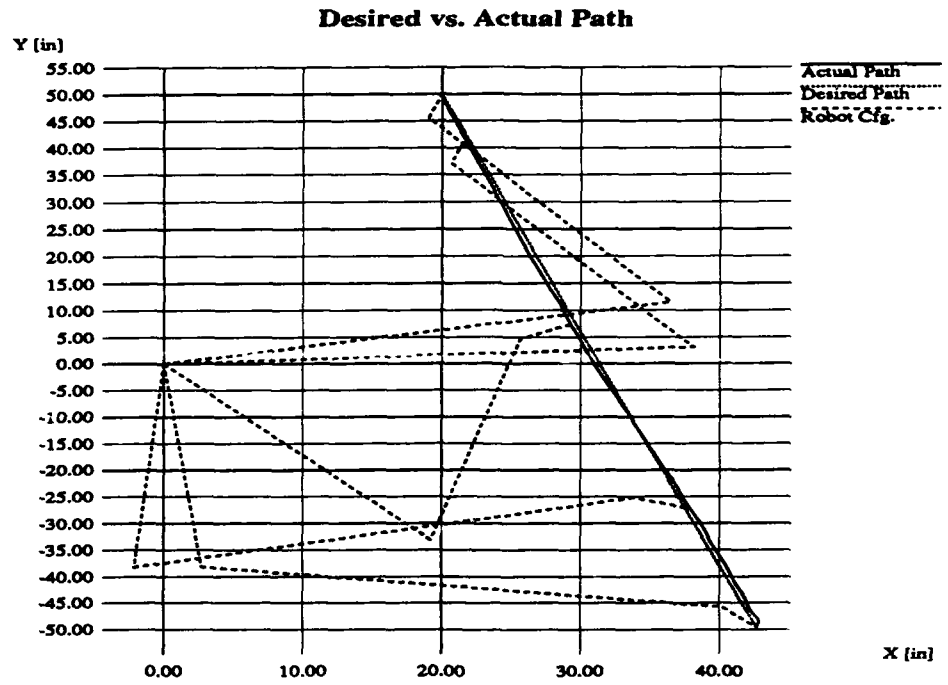
Initial position: $\theta_0 = [-2.0, 1.7, 0.3]^T$; Final position: $x_{end} = [25.163, 46.060]^T$.

Figure 34 3-link robot - Example 1: the error



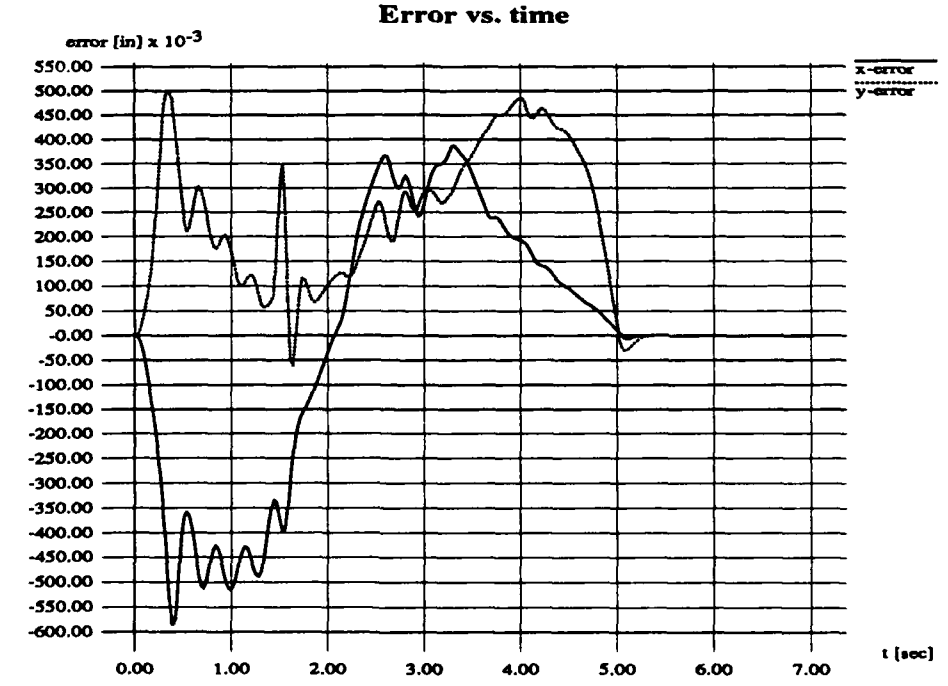
Maximum error:	[0.3932, 0.4603]	|.| = 0.5107
Error integral:	[1.0748, 1.1573]	|.| = 1.6467
Settling time error:	[0.0098, 0.0106]	|.| = 0.0145
Final error:	[0.0000, 0.0000]	|.| = 0.0000

Figure 35 3-link robot - Example 2: the path



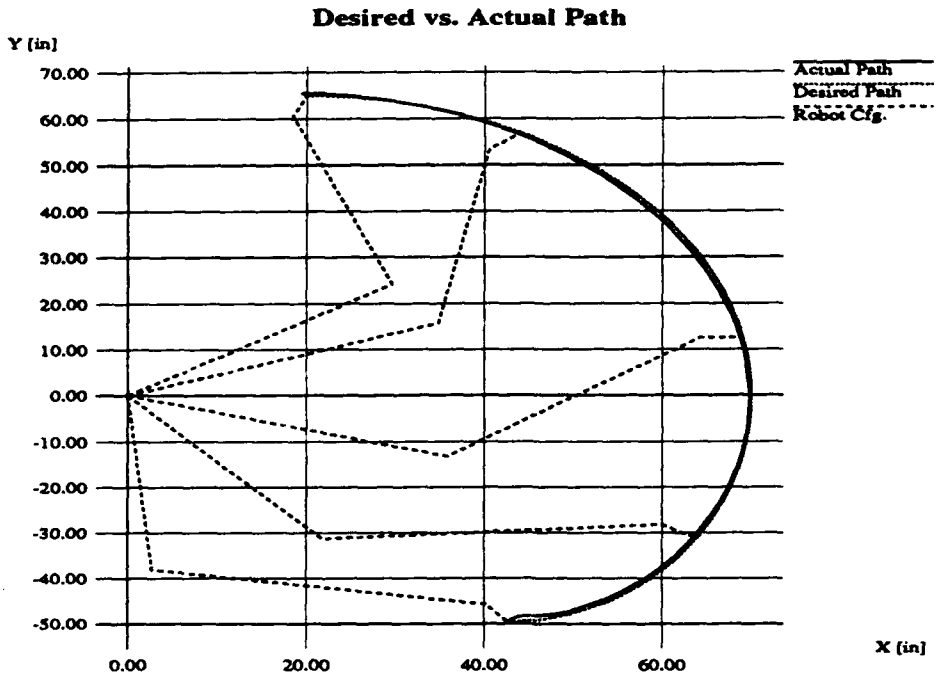
Initial position: $\theta_0 = [-1.5, 1.3, -0.8]^T$; Final position: $x_{end} = [20.0, 50.0]^T$.

Figure 36 3-link robot - Example 2: the error



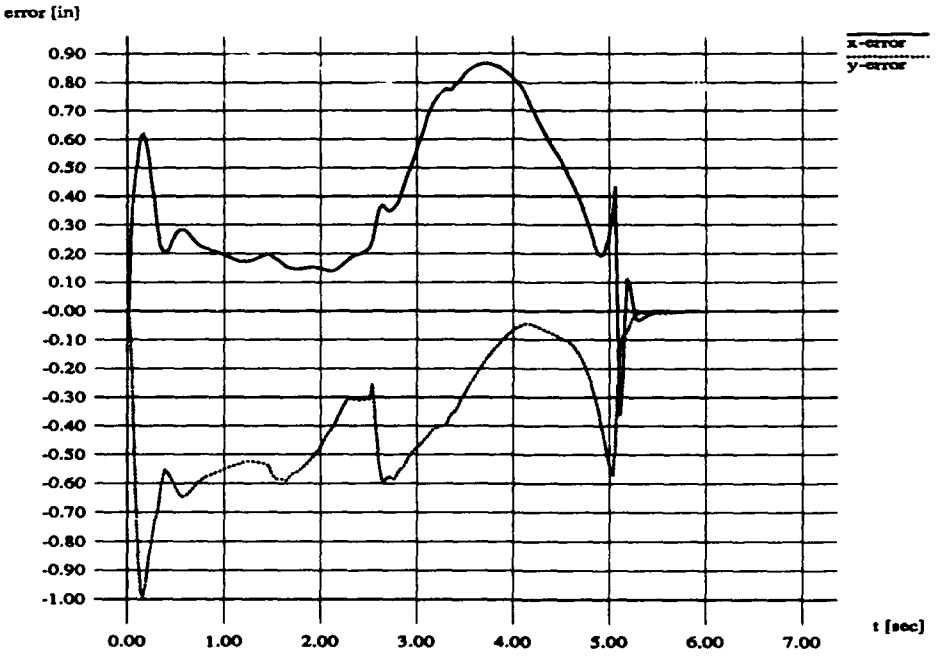
Maximum error:	[0.5851, 0.5021]	|.| = 0.7577
Error integral:	[1.2982, 1.2855]	|.| = 1.9512
Settling time error:	[0.0084, 0.0067]	|.| = 0.0107
Final error:	[0.0000, 0.0000]	|.| = 0.0000

Figure 37 3-link robot - Example 3: the path



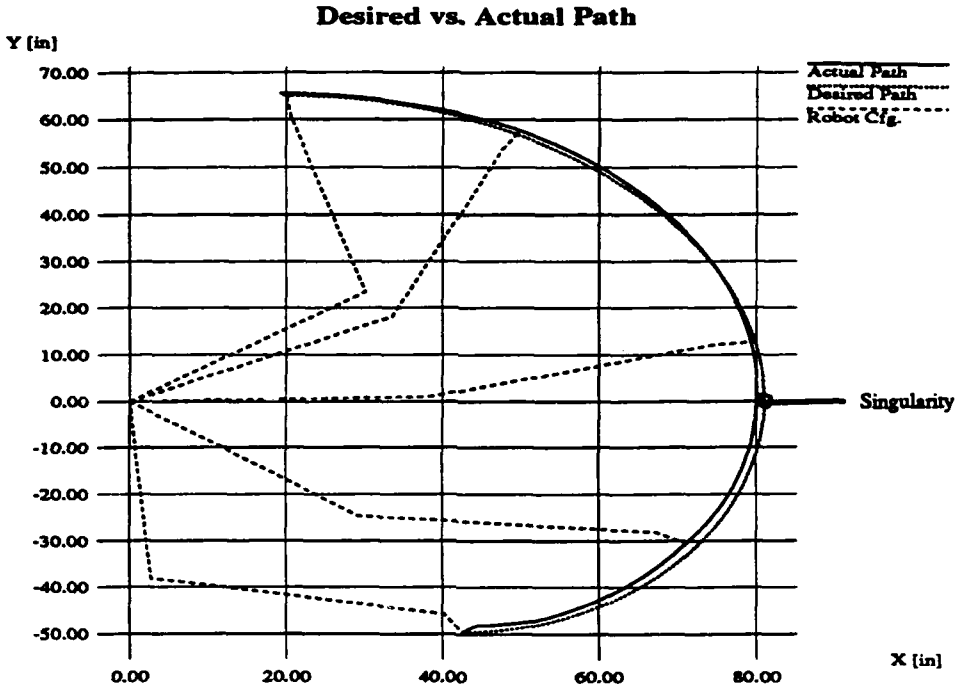
Initial position: $\theta_0 = [-1.5, 1.3, -0.8]^T$; Interm.pos. $x_{int} = [70.0, 0.0]^T$, Final pos.: $x_{end} = [20.0, 65.0]^T$.

Figure 38 3-link robot - Example 3: the error



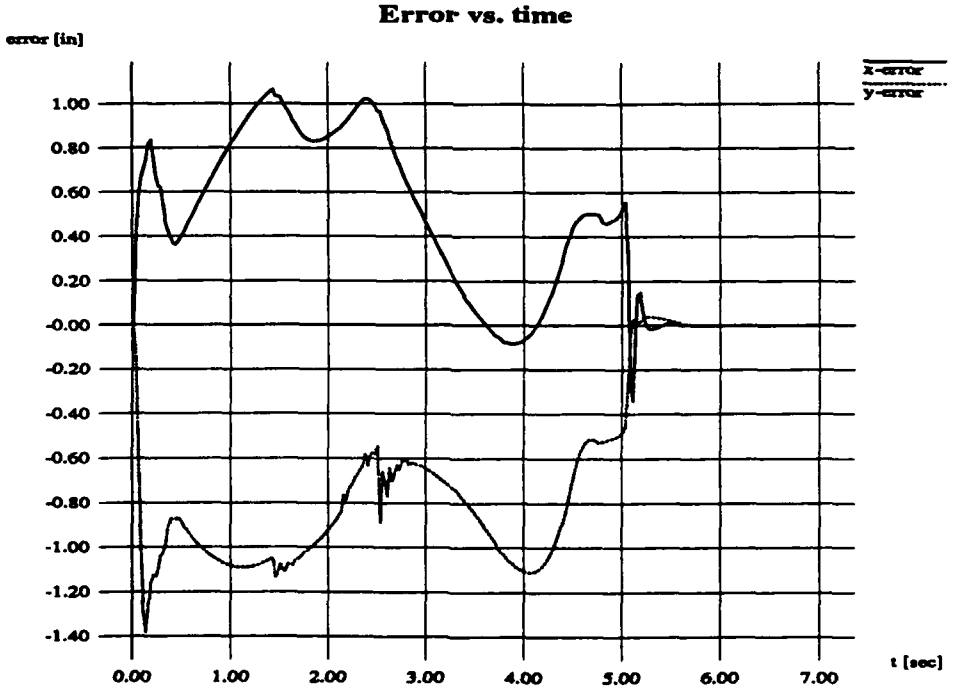
Maximum error:	[0.8658, 0.9919]	|.| = 1.1694
Error integral:	[2.1001, 2.0875]	|.| = 3.2846
Settling time error:	[0.2922, -0.5515]	|.| = 0.6241
Final error:	[0.0000, 0.0000]	|.| = 0.0000

Figure 39 3-link robot - Example 4: the path



Initial position: $\theta_0 = [-1.5, 1.3, -0.8]^T$; Interm.pos. $x_{int} = [81.01, 0.0]^T$, Final pos.: $x_{end} = [20.0, 65.0]^T$.

Figure 40 3-link robot - Example 4: the error



Maximum error:	[1.0627, 1.3830]	|.| = 1.5679
Error integral:	[2.8710, 4.3914]	|.| = 5.4744
Settling time error:	[0.5307, -0.4808]	|.| = 0.7161
Final error:	[0.0000, 0.0000]	|.| = 0.0000

Discussion

The simulation results show that the scheme works equally well for non-redundant and redundant robots. As verified in the graphs on pages 36 to 43, the errors for a robot with two links are approximately of the same magnitude as for a robot with three links, where the robots, with total link lengths of 80.0 in and 81.01 in, have comparable link lengths.

Dependency on the controller frequency

Referring to Table 10 (p. 34), Table 11 (p. 34), Table 12 (p. 44) and Table 13 (p. 45) and many other experiments conducted, all errors decrease when using a higher controller frequency. (This is only valid for the generally used scheme with oscillation detection and *dx*-input filtering.) The observation can most likely be attributed to the *dx*-input filter and not the smaller stepsize, as one might expect at first thought. Smaller stepsizes involve smaller errors per step. However, if the correction of an error causes another error which is larger than the one to be corrected, this process might support itself for a while and result in large position errors.

Oscillation detection on its own is only in part helpful, as can be seen in the mentioned tables. Going to higher frequencies increases the maximum and the integral error for some trajectories. Only the scheme with *dx*-input filtering improves both errors for almost any increase of the frequency.

As in Table 10 (p. 34), if oscillation occurred, the result for maximum error is marked with a 'star' *. Very light oscillation is marked with a 'plus' $^{+}$. Very light oscillation only during the last two seconds (servoing to target position, desired velocity is zero) is marked with a 'minus' $^{-}$.

Table 12 Dependency of the maximum tracking error on the controller frequency

trajectory

controller frequency [Hz]	*dx*-input filter OFF			*dx*-input filter ON		
	1	2	3	1	2	3
40	2.1070*	0.8723	13.4820*	0.6762^{+}	0.9173	1.5679^{+}
50	0.5623^{+}	0.8069^{-}	18.3842*	0.5788^{+}	0.8522	1.3103^{-}
60	0.5209^{-}	0.7549	30.8794*	0.5284^{+}	0.7929	1.1606^{-}
80	0.4622^{+}	0.9219^{+}	24.7492*	0.4647^{+}	1.2724^{+}	0.9609^{-}
120	0.3901	0.5824^{+}	30.8355*	0.3914	0.6115^{+}	0.8115^{-}

Maximum Error

Table 13 Dependency of the integral error on the controller frequency

trajectory

controller frequency [Hz]	*dx*-input filter OFF			*dx*-input filter ON		
	1	2	3	1	2	3
40	2.2995	2.2479	14.0296	2.0092	2.2707	5.2561
50	1.7484	2.0378	13.2070	1.7479	2.0407	4.4166
60	1.5680	1.8804	14.3729	1.5720	1.8797	3.8923
80	1.3048	1.6383	12.9900	1.3014	1.7244	3.3162
120	0.9087	1.2358	10.3518	0.9102	1.2219	2.7139

Integral Error

Dependency on configuration

There are configurations where the performance of the scheme degrades. This can be seen from the analysis in section 3.2.3 on page 27. Coming close to singular positions, the scheme tends more to produce errors. This does not mean, however, that the scheme fails completely, it is just that locally the error is large. Except for problems that are treated right after this, the scheme tries to avoid singular positions, unless it has to pass through them, as can be seen in the next two plots and in Figure 37 (p. 42).

Figure 41 2-link robot passing singular position

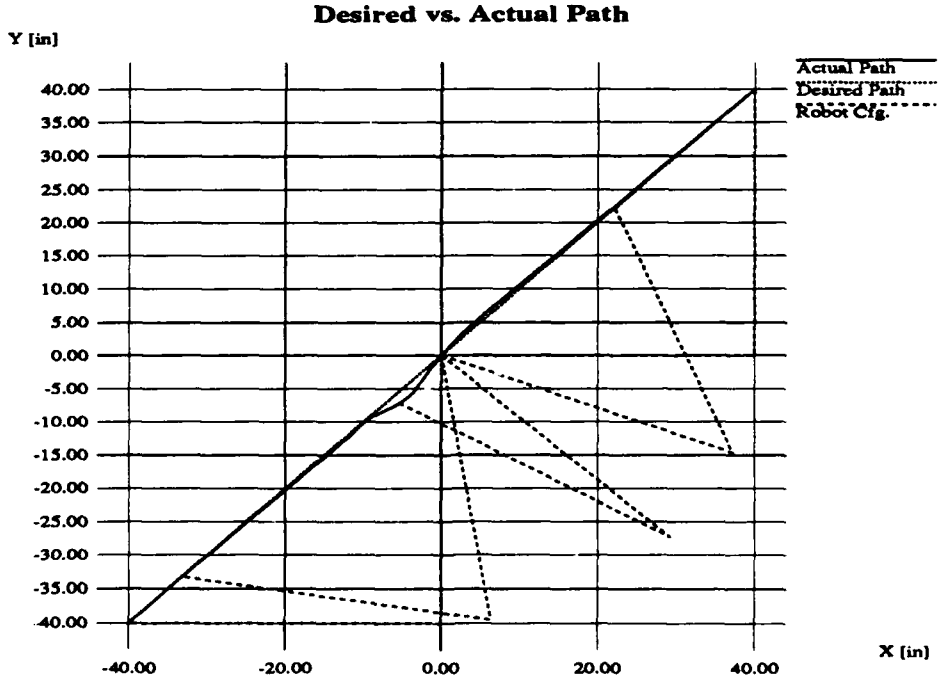
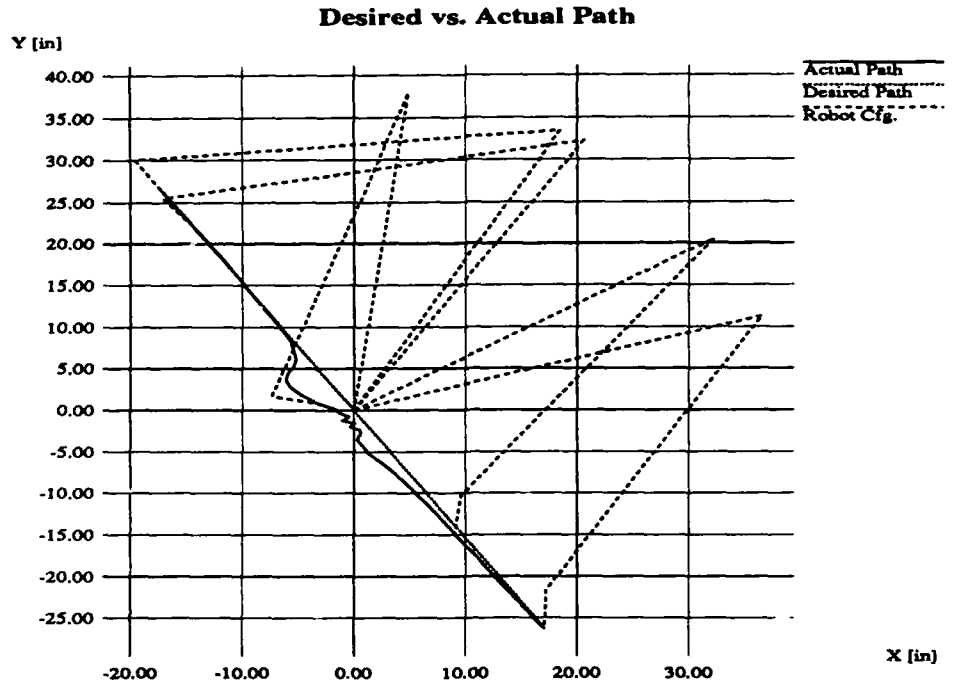


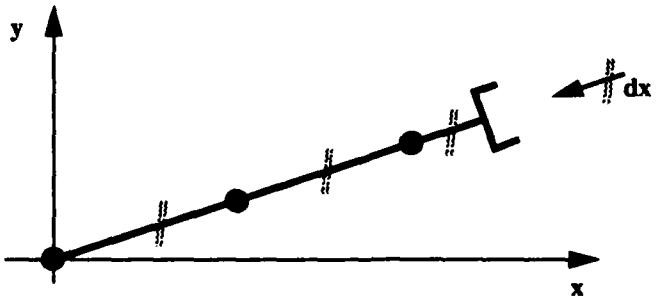
Figure 42 3-link robot passing singular position



Since the scheme works locally, large joint movements to avoid singular positions are very unlikely. In Figure 42, the robot could move link 1 and 2 to the other side of the reference path (clockwise) to avoid a singular position. Our scheme however doesn't favor such kind of trajectory and passes through the singular position.

There is one kind of configuration which causes problems. If all links are aligned, and the desired *dx* is exactly in direction of the links, no movement at all will occur (see illustration, Figure 43).

Figure 43 Problematic configuration with all links aligned



We tried to solve this problem by introducing small movements where no movement should occur. This worked, however the integral and the maximum errors rose by a not negligible amount. Since this special position is very improbable (at least in combination with the desired path), and since in real applications measurement errors will introduce

random correction movements in different directions, we felt that the disadvantageous feature weighs less than the generally increased error, and we didn't maintain the changes any further.

Other membership functions

In the simulation version the membership functions were designed for unit link length, and the parameters passed to the FLC were scaled by the maximum link length to compensate for this, which made it possible to change the robot geometry without changing the membership functions.

However, the optimal Fuzzy Mapping result is achieved only for the link with maximum length. In the 3-link-simulation, the first two links were of same length, and link 3 was significantly smaller. We therefore tried to design membership functions which would compensate for the scaling.

The results, though, didn't differ significantly from the original scheme. Some results even showed a weaker performance. Therefore, and to maintain the independence from link lengths, and thus the ability to change the robot geometry without changing the membership functions, we stayed with the original scheme.

Prediction of the error

We examined the relation between the tracking error and the norm of the difference between the Jacobian J and the effective Jacobian J_F resulting from our Fuzzy scheme:

$$e \sim ||J - J_F|| \qquad \text{(EQ 3-34)}$$

J_F was defined as the pseudo-inverse to J_F^{-1}, which could be generated from the equation

$$d\theta = J_F^{-1} dx \qquad \text{(EQ 3-35)}$$

For simplicity, lets call

$$J_F^{-1} = F \qquad \text{(EQ 3-36)}$$

Then

$$d\theta = F dx \qquad \text{(EQ 3-37)}$$

$$J_F d\theta = J_F F dx$$

but the following must hold as well:

$$J_F d\theta = dx \qquad \text{(EQ 3-38)}$$

Therefore, we can require

$$J_F F = I \tag{EQ 3-39}$$

and conclude

$$J_F = (F^T F)^{-1} F^T \tag{EQ 3-40}$$

Using what we have obtained now, we have compared the original and the Fuzzy-mapping-derived, effective Jacobian and used the norm of the difference for checking on any relation with the position error:

$$||J - J_F|| \overset{?}{\sim} e \tag{EQ 3-41}$$

Lets call the quantity on the left hand side of above equation "Jacobian difference norm" or short "JDN". The norm we used was the infinity norm:

$$||A|| = ||A||_\infty = \max_i \left(\sum_{j=1}^{n} |a_{ij}| \right) \tag{EQ 3-42}$$

The relations we tested for were:

- ratio of error and JDN (see sample plots in Figure 44 to Figure 46)
- error plot compared to JDN plot
- configurations at minima and maxima of ratio of error and JDN
- largest JDN and configuration with largest position error

We couldn't find any significant relations, which would allow us to predict the error or have some other measure for the error, in order to take precautions and try to keep the error small.

Figure 44 Ratio of error and JDN for random trajectory

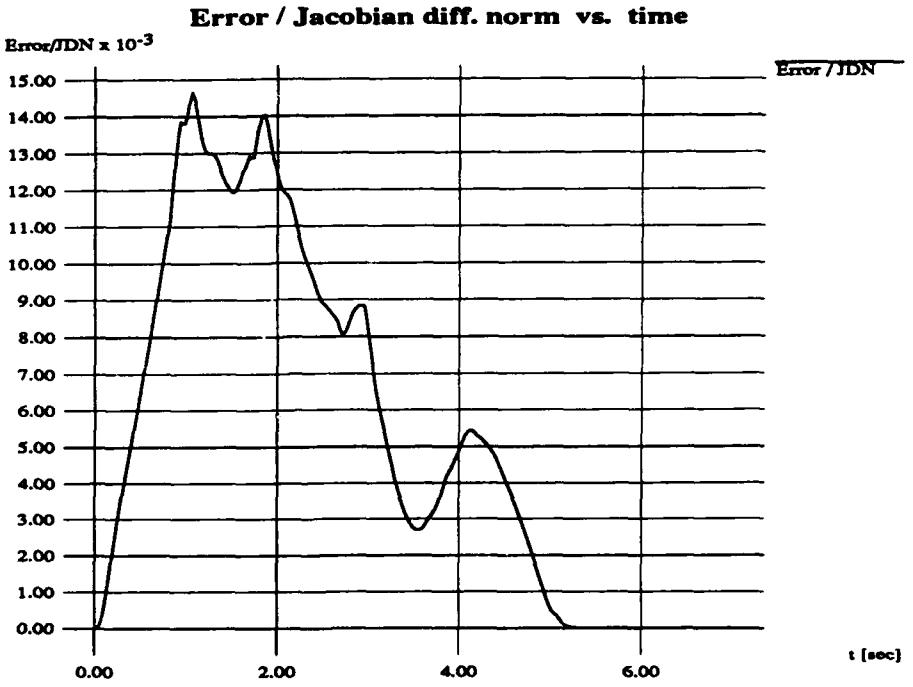


Figure 45 Ratio of error and JDN for random trajectory

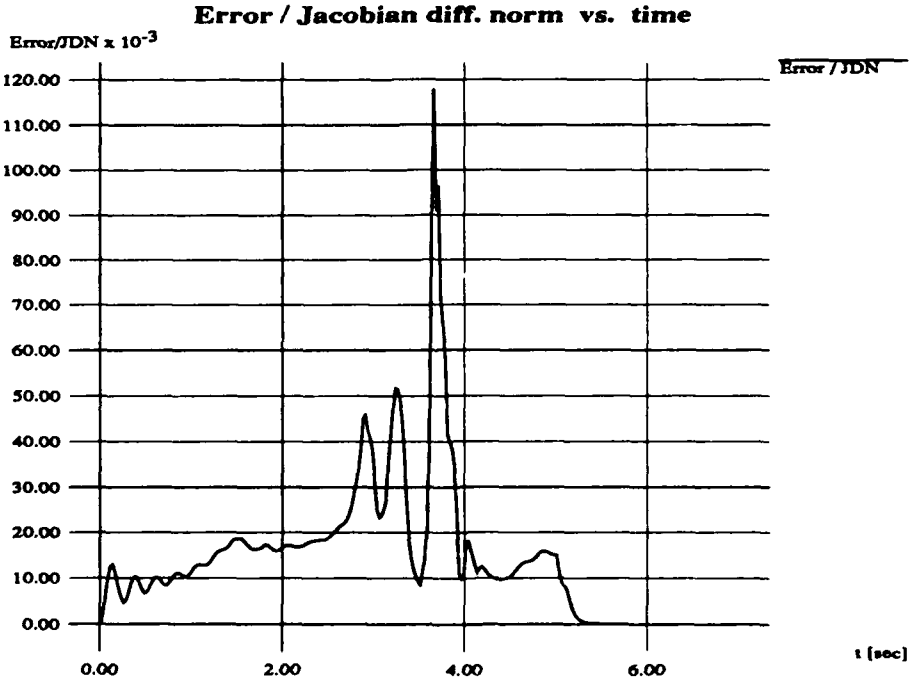
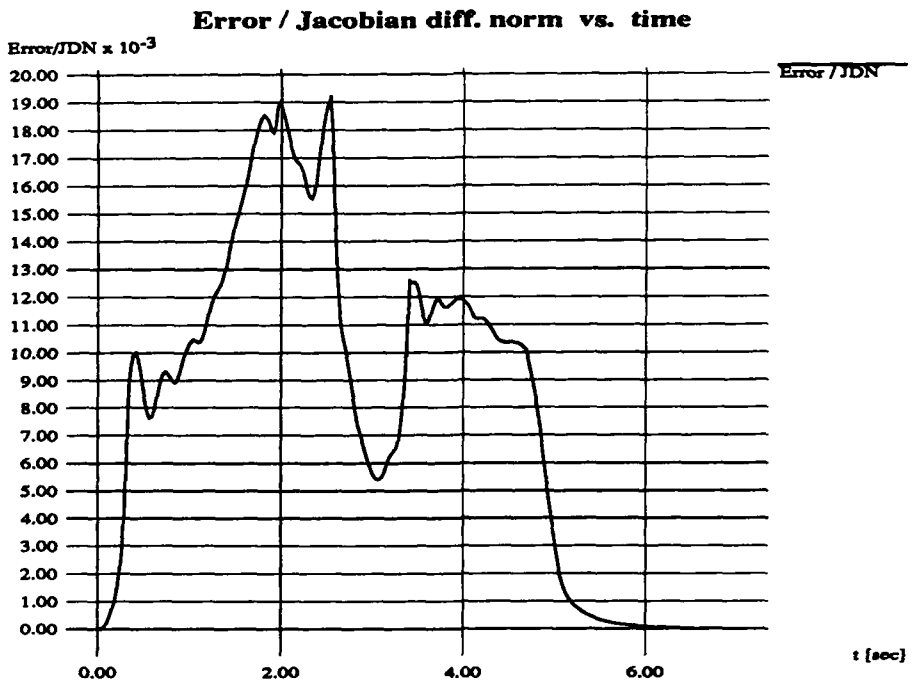


Figure 46 Ratio of error and JDN for random trajectory



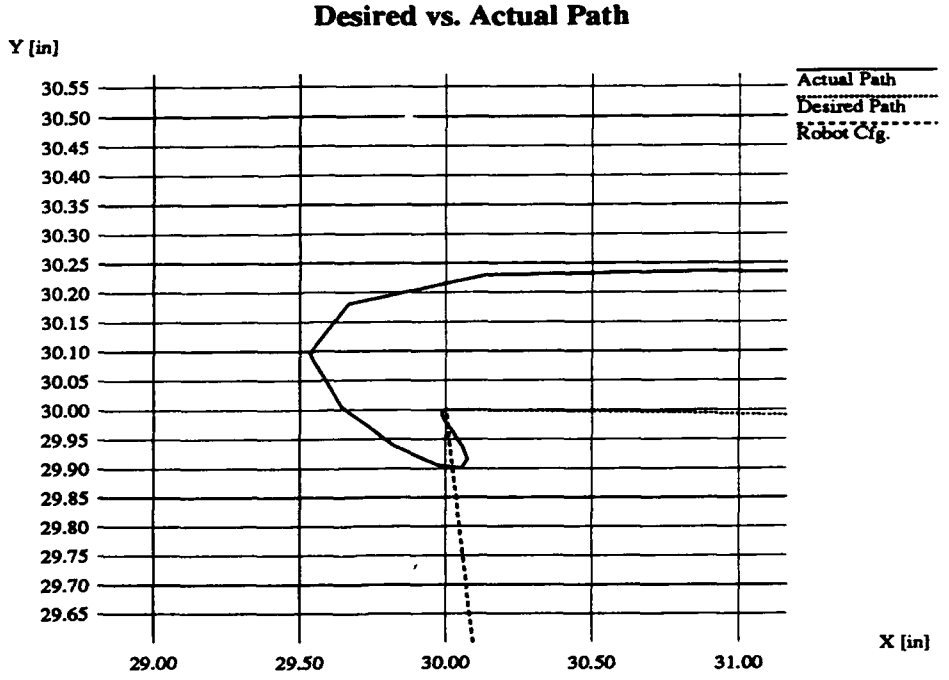
Steady state error

The steady state error in general is zero. This is an interesting feature of this scheme, and the explanation is simple. Even when the scheme is a little off, it basically goes in the right direction. If $t > t_{set}$, we are servoing constantly to the same (final) position. Thus, with each step, we are getting closer to the target, till we finally hit it.

The following is a zoom-in in one of the simulation results presented earlier (Figure 31 (p. 39)).

We can see that the mapping tends much too much to the right of the desired dx, but finally, we come close enough to hit the target.

Figure 47 Behavior of the Inverse Kinematic Mapping at the target position (here: 2-link-robot)



Typically, it takes approximately 0.5 sec (at 40 Hz) after t_{set} to come to the target position.

3.4 Implementation for teleoperation

3.4.1 Introduction

To test the performance of our method in a real-time environment, we used this Fuzzy Inverse Kinematic Mapping for the teleoperation of SM^2. The input device is a "flying mouse", the **bird**, which provides five degrees of freedom: three in position, and the angles α and γ for orientation.

Since our goal was the implementation of a scheme that is capable of dealing with input with six degrees of freedom, the remaining orientation parameter β has been set to zero, which complemented the bird output data.

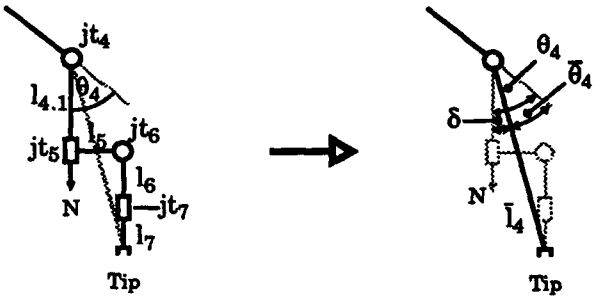
3.4.2 Analysis

1. Three-link robot

In this case, the algorithm that was studied in simulation was transferred directly to the robot. We chose joint 2, 3 and 4 to be operated, while the remaining joints 1, 5, 6 and 7 were kept fixed. This resulted in a three-link robot working in a vertical, two-dimensional plane, so indeed we were using a redundant planar robot.

Instead of using link 4 as it is, we treated the whole (fixed) chain from link 4 to link 7 together as a new link 4, which then actually has been controlled. This is clarified in the following Figure 48.

Figure 48 Treatment of the chain from link 4 to link 7 as new link 4



The chain from link 4 to link 7 (with fixed joint jt_5 to jt_7) is replaced by a single link $\bar{l}_4$. Accordingly, joint angle θ_4 is reduced by the amount of δ.

For the operation of the robot, the new, effective $\bar{l}_4$ and $\bar{\theta}_4$ have been used, which can readily be derived as:

$$\delta = \text{atan2}\,(l_5, l_{4.1} + l_6 + l_7) \qquad \text{(EQ 3-43)}$$

$$\bar{\theta}_4 = \theta_4 - \delta \qquad \text{(EQ 3-44)}$$

$$\bar{l}_4 = \sqrt{l_5^2 + (l_{4.1} + l_6 + l_7)^2} \qquad \text{(EQ 3-45)}$$

For our robot, the numerical values are $\delta = 0.19645\,[\text{rad}]$ and $\bar{l}_4 = 16.548\,[\text{in}]$.

A x_F-y_F-coordinate system, which has been used for the algorithm and also for the result printouts, was attached to the plane in which the robot could move.

The bird movement has been projected in this plane by rotation around θ_1, i.e., the radial distance from the base and the elevation above the x-y-plane have been taken from the bird data, while θ_1 was kept at its constant value.

Using the simplified model in Figure 49, where $\bar{l}_4$ is replaced by l_4 and $\bar{\theta}_4$ by θ_4, the forward kinematics and the Jacobian can be derived as follows:

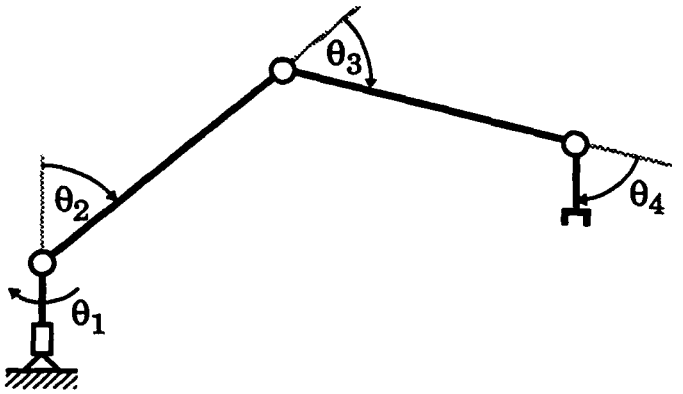
$$\begin{bmatrix} x_F \\ y_F \end{bmatrix} = \begin{bmatrix} l_2 s_2 + l_3 s_{23} + l_4 s_{234} \\ l_2 c_2 + l_3 c_{23} + l_4 c_{234} \end{bmatrix} \qquad \text{(EQ 3-46)}$$

$$\begin{bmatrix} dx_F \\ dy_F \end{bmatrix} = \begin{bmatrix} l_2c_2 + l_3c_{23} + l_4c_{234} & l_3c_{23} + l_4c_{234} & l_4c_{234} \\ -l_2s_2 - l_3s_{23} - l_4s_{234} & -l_3s_{23} - l_4s_{234} & -l_4s_{234} \end{bmatrix} \begin{bmatrix} d\theta_2 \\ d\theta_3 \\ d\theta_4 \end{bmatrix} \quad \text{(EQ 3-47)}$$

In these equations, the following abbreviations have been used:

$$c_{i\ldots k} = \cos(\theta_i + \ldots + \theta_k)$$
$$s_{i\ldots k} = \sin(\theta_i + \ldots + \theta_k) \quad \text{(EQ 3-48)}$$

Figure 49 Simplified model of 3-link robot in vertical plane



2. Seven-link robot

Now, the operation of the bird affected all seven joints of the robot. The forward kinematic expression and the Jacobian have been derived explicitly, but are omitted here because of the size. The location vector x was now comprised of the position and orientation:

$$x = [x\ y\ z\ \alpha\ \beta\ \gamma]^T \quad \text{(EQ 3-49)}$$

The angles α, β and γ are related to the joint angles by:

$$\alpha = \frac{3}{2}\pi - (\theta_2 + \theta_3 + \theta_4 + \theta_5) \quad \text{(EQ 3-50)}$$

$$\beta = \theta_1 + \theta_5 \quad \text{(EQ 3-51)}$$

$$\gamma = \theta_7 \quad \text{(EQ 3-52)}$$

Therefore, the coefficients c_{ij} (the entries of the Jacobian) for the Cartesian angles α, β and γ are ± 1 or 0:

$$c_{\alpha 2} = c_{\alpha 3} = c_{\alpha 4} = c_{\alpha 5} = -1 \quad \text{(EQ 3-53)}$$

$$c_{\beta 1} = c_{\beta 5} = 1 \quad \text{(EQ 3-54)}$$

$$c_{\gamma 1} = 1 \quad \text{(EQ 3-55)}$$

and the rest of the coefficients $c_{\alpha j}$, $c_{\beta j}$, $c_{\gamma j}$ which are not defined above are zero.

Since the coefficients for x, y, z have not been scaled (to save a little computation time), we had to multiply the coefficients for α, β and γ by some factor in the order of the largest link length (38.23 in), which had been used for scaling.

This gave us also the possibility to express priorities. For example, for α, $c_{\alpha 6} = -100$, while $c_{\alpha 2} = -10$, $c_{\alpha 3} = -20$ and $c_{\alpha 4} = -30$. The effect is that joint 6 handles more of the orientation part, while joints 2 to 4 handle mainly the position components *dx*, *dy* and *dz*.

3.4.3 Results

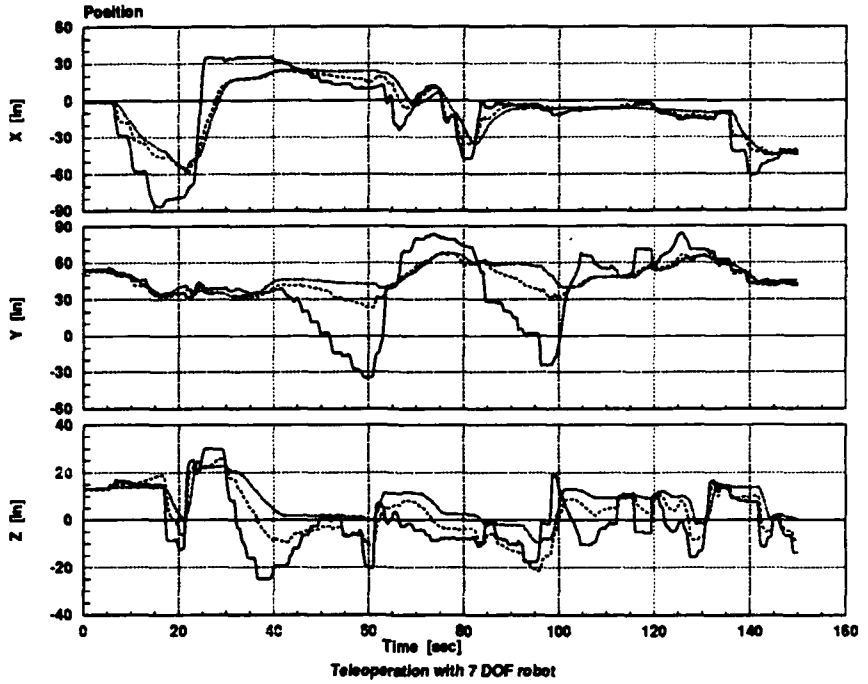
We found that the Inverse Kinematics module for the 3-link robot experiment was not able to run at 40 Hz. A performance analysis revealed that most of the time was spent in evaluating the membership functions and deriving an output from the rule-base in the FLC. This required approximately 80% of the total Inverse Kinematics processing time. The computation of the elements of the Jacobian and the additional processing like weighing and filtering, which takes place in `InvkinCycle`, requires about 15% of the total time. The computation of the sines and cosines (`sincos()`) and the square roots (`sqrt()`) and all other functions used (`atan2()`, `fabs()`, type conversions, ...) required less than 5% of the total time.

So, in order to speed the computation up, we made two minor changes. The data type used for real numbers throughout `libfuzzy.c`, which contains the functionality of the FLC, was set to `double`. Secondly, the way `evaluate_mf` checks different membership function types was rearranged. Both changes speeded computation up significantly.

Finally, we ran the whole application at 30 Hz. The *dx*-input filter time constant was left at a value of $c = 0.40$. The algorithm was found to be working satisfactorily.

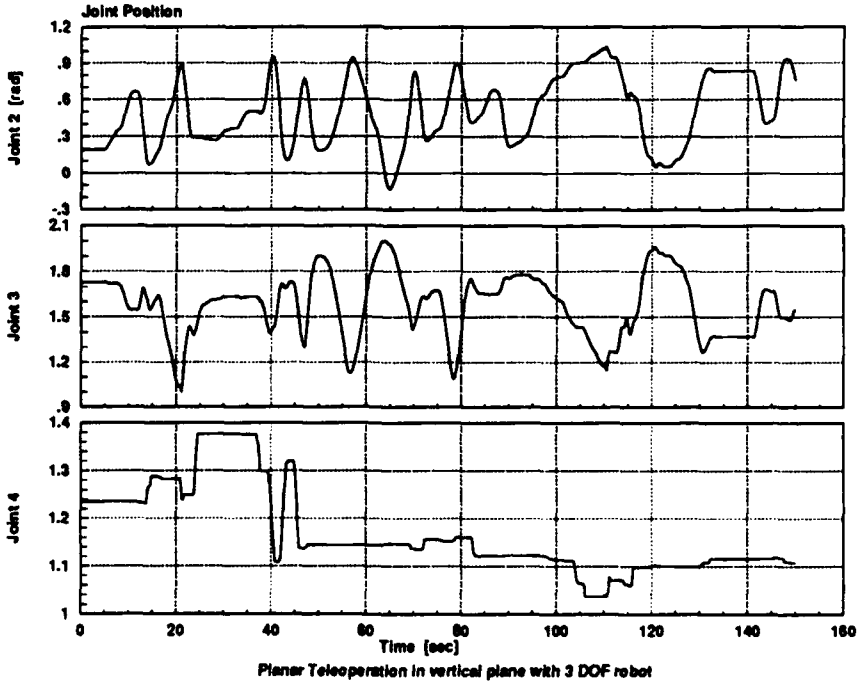
A sample teleoperation result is shown in the following figures.

Figure 50 Sample 3-dof teleoperation: Cartesian position



The position refers to the coordinate system attached to the vertical plane (see text). The results show the reference position as read from the bird (solid line), the commanded position from the Inverse Kinematic Mapping (dashed line), and the measured robot position (dotted line).

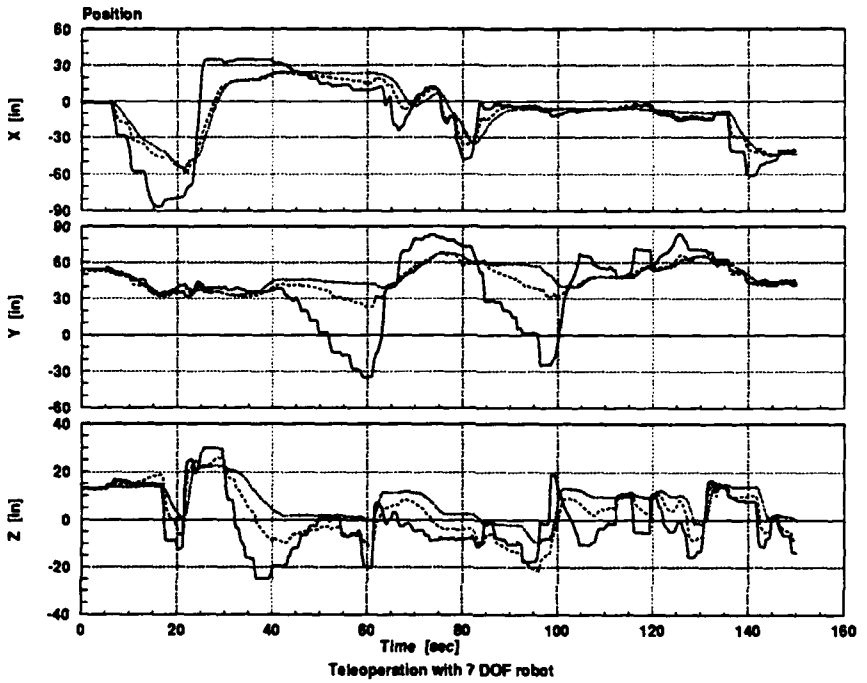
Figure 51 Sample 3-dof teleoperation: Joint position



Robot with 7 links

Here, the FLC was obviously even more a bottle-neck, since more computation was required: the number of joints rose from three to seven, and the Cartesian coordinates from two to six. However, running the application at 15 Hz produced satisfactory results.[2] Plots of experimental results follow.

Figure 52 Sample 7-dof teleoperation: Cartesian position (example 1)



3.4.4 Discussion

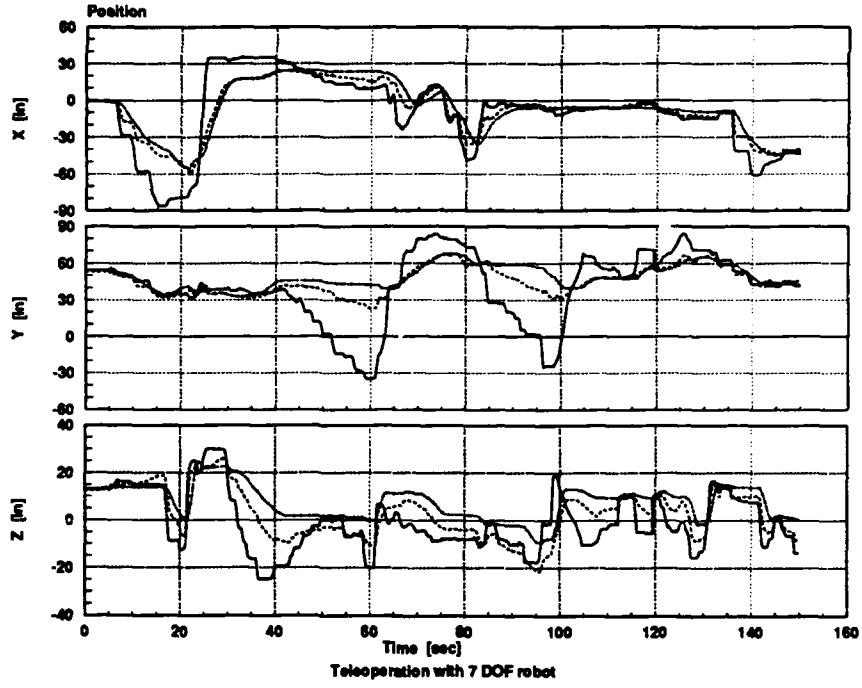
In judging Figure 52 and Figure 53, two points should be considered. First, the *bird* produces very noisy data. Secondly, the conventional Inverse Kinematics that had originally been used deteriorated considerably when the robot came close to a position where link 2 and 3 were aligned (stretched-out position). This led to jerky motion.

This problem could be reduced considerably by using the Fuzzy Inverse Kinematic Mapping. The robot can be operated safely at positions close to stretched-out. However, trying to stretch the robot completely excites oscillations at the robot tip. In general, the movements of the robot were very smooth, which partly can be attributed to the filtering we perform.

As can be seen in Figure 52 and Figure 53, the movements in x-direction were significantly better than in y- and z-direction. This is mainly caused by the gravity compensation system, as has been observed also in other experiments.

2. It should be considered that the bird module can be run at no higher rate than 20 Hz.

Figure 53 Sample 7-dof teleoperation: Cartesian position (example 2)



The low control rates used are not a disadvantage. At higher rates, the robot behaves less smooth. This is intuitive, since the robot is a mechanical lowpass system, and there is some delay between setting a reference position and the time when the robot reaches it. Thus, the controller receives large errors as input while the robot is still moving to the desired position, and produces large control signals. This leads to further acceleration of the robot, which makes it more prone to overshoot. At lower rates, however, the robot has more time and comes closer to the commanded position before a new control cycle starts, which means that the control signals will be smaller and smoother compared to the previous case.

As for the time spent in the FLC, this time might be reduced by several means. First, since the time is directly proportional to the number of rules, a smaller number of rules, which might have a comparable performance, might do as well. Secondly, once the membership functions and the rule-base have been determined, the whole FLC might be converted into a look-up table. This would speed up the determination of the Fuzzy output considerably, at the expense of a little less accuracy (not very important, since the Fuzzy scheme works with vague concepts anyway), and, very important, also at the expense of a large memory required. The third option would be to implement the FLC to work with integers internally and to scale real-valued values (input to and output from FLC, membership functions) appropriately.

3.5 Summary

We have developed an Inverse Kinematic Mapping based on Fuzzy Logic. This method does not require any constraints to be imposed on the robot configuration, nor does it require any functions to be optimized. It works identically for non-redundant and for redundant robots. The solution is obtained in closed-form, as opposed to numerical solutions of popular schemes. We have shown the usability of the method by employing it for real-time teleoperation of a seven-link robot in the six-dimensional Cartesian space.

4.0 Acknowledgements

The SM^2 testbed research is supported by Space Project Office of Shimizu Corp., Japan. We would like to thank the following colleagues for their support and stimulating discussions and for revising drafts of this report: Todd Newton, Pradeep Khosla, Ben Brown, Randy Casciola and Veda Young. The first author's visit at CMU was supported by Cusanuswerk.

5.0 Bibliography

[1] C. von Altrock and H.-J. Zimmermann. Wissensbasierte Systeme und Fuzzy Control. *Sonderdruck aus dem Informatik-Themenheft der Rheinisch-Westfälischen Technischen Hochschule Aachen*; 1/1991; pp. 1-6

[2] B. Armstrong. Friction: experimental determination, modeling and compensation. In *Proceedings of the 1988 IEEE International Conference on Robotics and Automation*; Philadelphia, PA, USA; April 1988; pp. 1422-1427

[3] H. Asada and J.-J.E. Slotine. *Robot Analysis and Control*. J. Wiley, 1986.

[4] J. Baillieul. Avoiding obstacles and resolving kinematic redundancy. In *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, San Francisco, California, USA; April 1986; vol.3, pp. 1698-1704

[5] C.C. de Wit and V. Seront. Robust adaptive friction compensation. In *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*; Cincinnati, OH, USA; May 1990; pp. 1383-1388

[6] Z. Cao, A. Kandel, L. Li and J. Han. Mechanism of Fuzzy Logic Controller. In *Proceedings of the First International Symposium on Uncertainty Modeling and Analysis*; College Park, MD, USA; Dec. 1990; pp. 603-607

[7] P.H. Chang. A closed-form solution for the control of manipulators with kinematic redundancy. In *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, San Francisco, California, USA; April 1986; vol.1, pp. 9-14

[8] C. Chevallereau and W. Khalil. A new method for the solution of the inverse kinematics of redundant robots. In *Proceedings of the 1988 IEEE International Conference on Robotics and Automation*; Philadelphia, PA, USA; April 1988; pp. 37-42

[9] C. Chevallereau and W. Khalil. Efficient method for the calculation of the pseudo inverse kinematic problem. In *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*, Raleigh, North Carolina, USA; March-April 1987; vol.3, pp. 1842-1848

[10] P.E. Dupont. Friction modeling in dynamic robot simulation. In *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*; Cincinnati, OH, USA; May 1990; pp. 1370-1376

[11] A. Ghosal and B. Roth. A new approach for kinematic resolution of redundancy. In *International Journal of Robotics Research*, vol.7, no.2; April 1988; pp. 22-35

[12] A. Gogoussis and M. Donath. Coulomb friction effects on the dynamics of bearings and transmissions in precision robot mechanisms. In *Proceedings of the 1988 IEEE International Conference on Robotics and Automation*; Philadelphia, PA, USA; April 1988; pp. 1440-1446

[13] J.M. Hollerbach and K.C. Suh. Redundancy resolution of manipulators through torque optimization. In *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, St. Louis, Missouri, USA; March 1985; pp. 1016-1021

[14] N. Imasaki and S. Nishida. Positioning control for a space manipulator system using fuzzy control method. FAX copy; Japan; pp. 57-60

[15] B. Kuipers and K. Åström. *The composition of heterogeneous control laws*. Technical Report AI90-138, Artificial Intelligence Laboratory; The University of Texas at Austin; Sept. 1990

[16] D.P. Kwok, P. Tam, C.K. Li and P. Wang. Analysis and design of fuzzy PID control systems. In *International Conference on Control '91*; Edinburgh, UK; March 1991; vol.2, pp. 955-960

[17] C.C. Lee. Fuzzy logic in control systems: fuzzy logic controller. Parts I and II. In *IEEE Transactions on Systems, Man and Cybernetics*, vol.20, no.2; March-April 1990; pp. 404-435

[18] N. Matsunaga and S. Kawaji. Hybrid Controller with Fuzzy and PD Control. FAX copy; Japan.

[19] Y. Nakamura. *Advanced robotics: redundancy and optimization.* Addison-Wesley, 1991.

[20] G. Schmidt. *Grundlagen der Regelungstechnik: Analyse und Entwurf linearer und einfacher nichtlinearer Regelungen sowie diskreter Steuerungen.* Springer Verlag, 1987.

[21] C.W. de Silva and A.G.J. MacFarlane. Knowledge based control with application to robots. In *Lecture Notes in Control and Information Sciences*; *123*. Springer Verlag, 1989.

[22] D. Simon. *Fuzzy Systems Tutorial.* Documentation of VASC Retreat, The Robotics Institute, Carnegie Mellon University, June 1992.

[23] Y. Xu, B. Brown, M. Friedman and T. Kanade. *Control System of Self-Mobile Space Manipulator.* The Robotics Institute, Carnegie Mellon University, 1991.

[24] Y. Xu, R.P. Paul and H.-Y. Shum. Fuzzy Control of Robot and Compliant Wrist System. In *IEEE International Conference on Industrial Applications*, 1991.

[25] L.A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. In *IEEE Transactions on Systems, Man and Cybernetics*, vol.SMC-3, 1973; pp. 28-44

Appendix: Fuzzy Logic and Fuzzy Control

Fuzzy Logic was developed by L.A. Zadeh in 1965. It is a mathematical framework for representing vague concepts or ***fuzzy values***, as in

- the weather is ***hot***
- there are ***many*** people in this room
- the car is going ***slow***

This is usually referred to as 'uncertainty representation'.

Fuzzy Logic also provides mechanisms for manipulating these fuzzy values, for example:

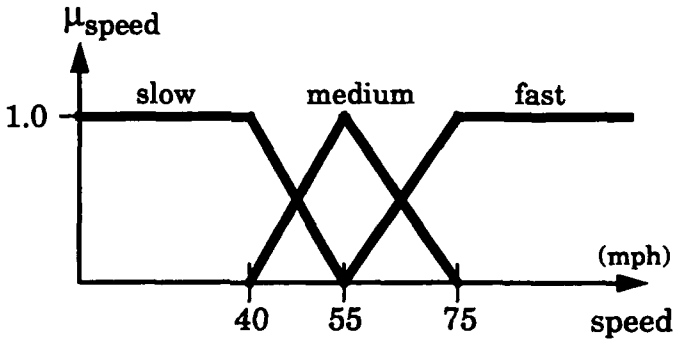
- If the weather is ***hot*** and a ***bit humid***, then wear ***light*** clothes (from [22]).
- If the house is ***attractive*** and ***well accessible***, it is ***very desirable***.

Membership function

One of the pivotal points in Fuzzy Logic is the representation of the fuzzy values by means of a membership function, μ_D: $u \rightarrow [0;1]$. Membership functions denote the degree $\mu_D(x)$ to which a value $x \in U$ can be represented by the fuzzy value D. Some authors refer to the $\mu_D(x)$ as the *appropriateness* for describing x with the descriptor D [15].

A fuzzy variable is a variable that can assume fuzzy values. For example, we might let the variable ***speed*** assume the fuzzy values ***slow***, ***medium*** and ***fast***. Sample membership functions are given in the following Figure A-1.

Figure A-1 Membership functions for *slow*, *medium* and *fast* for the fuzzy variable *speed*



Referring to the definitions in Figure A-1,

- the degree of membership of 70 mph to the concept *fast*, $\mu_{fast}(70)$, is 0.75.
- $\mu_{slow}(70) = 0$, i.e. the concept *slow* is completely inappropriate for describing a speed of 70 mph.

- combining the degrees of membership to all defined values, a speed of $v_{70} = 70$mph might be represented as

$$v_{70} = 0.00/slow + 0.25/medium + 0.75/fast, \qquad \text{(EQ 1-1)}$$

where '*x*/*y*' is read as 'the appropriateness for using *y* as descriptor is *x*', and '+' is read as '*and*'.

Note: Membership functions do not have to add up to 1.0 at any one point. They also do not have to be linear functions. We only favor linear functions because their representation on a computer is simple.

A Fuzzy Logic Controller (FLC) consists of a collection of Fuzzy rules of the form

IF (*in1* = *u*) **AND** (*in2* = *v*) **THEN** (*out* = *w*),

where *in1*, *in2* and *out* represent fuzzy variables, and *u*, *v* and *w* are fuzzy values.

For example:

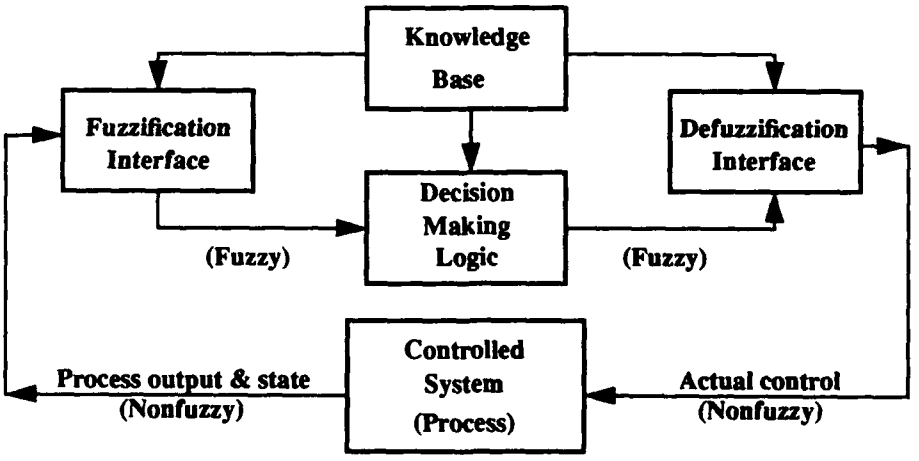
IF (*error* = ***large***) **AND** (*velocity* = ***small***) **THEN** (*torque* = ***large***).

The collection of these rules is called **rule-base.**

What is further needed for a FLC are mappings to convert crisp input variables (process output and state) into fuzzy variables (*fuzzification interface*) and to convert a fuzzy output to a crisp control value for the controlled system (*defuzzification interface*).

The basic configuration for an FLC is show in Figure A-2 [17].

Figure A-2 Basic configuration of a Fuzzy Logic Controller (FLC)



The effect of a fuzzy rule

We have seen that a FLC works on a collection of rules which relate fuzzy input and output variables. How dos a FLC now *use* these rules in determining the output?

There are two mechanisms working. First, all rules work in parallel, and all outputs of all rules are combined to form the output of the FLC. This is described in the next section. Secondly, the output of a single rule is determined by the fuzzy value of the *consequent* and by the *weight* of the rule, which describes how appropriate it is to apply this rule.

The weight α of a rule is a combination of the membership function values of the antecedents of the rule (i.e. of the input variables). The two operators that are mainly used for this combination are:

- intersection: $x \wedge y = \min\{x, y\}$
- algebraic product: $x \cdot y = xy$

If we use the sample rule on page A-2, we can write

$$\alpha = \mu_u(in1) \wedge \mu_v(in2)$$

or

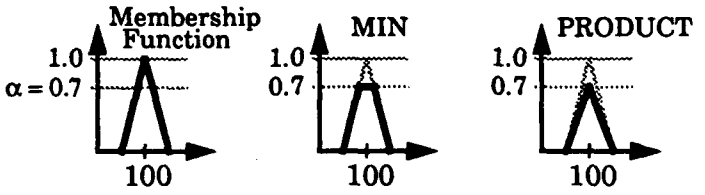
$$\alpha = \mu_u(in1) \cdot \mu_v(in2) \qquad \text{(EQ 1-2)}$$

depending on the operator we are using.

The weight α describes how much the rule applies. If both input values exactly match the fuzzy values ***u*** and ***v***, we assume that the rule should apply fully, and α will be exactly $\alpha = 1$. If one or both of the conditions are not met ($\mu_x(iny) = 0$), the rule shouldn't be used in determining the output of the FLC, and accordingly $\alpha = 0$. Between these extreme situations, the weight will assume values in the range from zero to one.

We now use α_i and combine it with the output value (consequent) w_i of the rule to form the rule's effective output value $\alpha_i \bullet w_i$. The same operators - intersection and algebraic product - are used here. The use of these operators in determining the effective output is demonstrated in Figure A-3.

Figure A-3 The effect of the weight α on the consequent of a rule depending on the used operator



Fuzzy Inference

Since we have determined the effective outputs $\alpha_i \bullet w_i$ of all rules, we now have to combine them to yield the overall output of the rule-base. This is usually performed using the union operator:

$$x \vee y = \max\{x, y\} \qquad \text{(EQ 1-3)}$$

Thus, the fuzzy result of the rule-base $out_{rulebase}$ can be written as

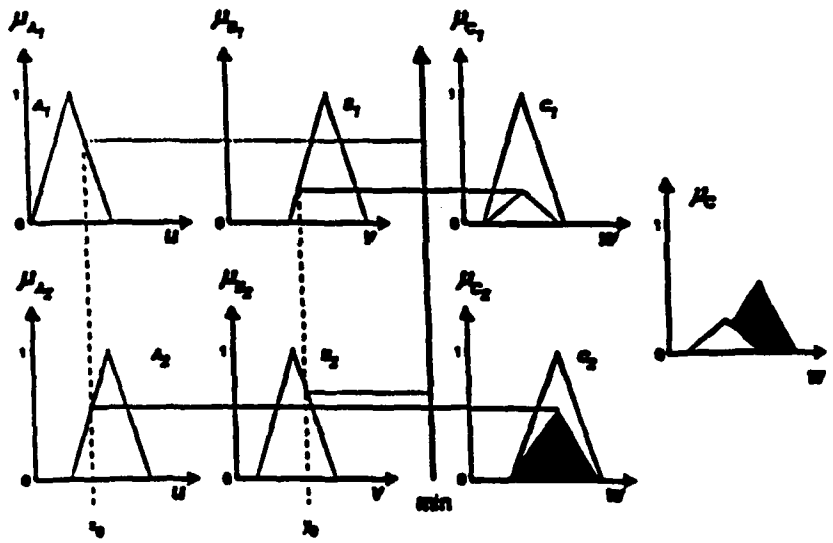
$$out_{rulebase}(x) = \max_i \{\alpha_i \bullet w_i(x)\} \qquad \text{for all } (x \in U) \tag{EQ 1-4}$$

Example

The example in Figure A-4 represents an FLC with two input variables, A and B, and one output variable, C. The rule-base consists of two rules. The effect of each rule is represented horizontally.

The membership functions for A are depicted on the left side: A_1 and A_2 might represent values like ***small*** and ***medium***, for example. The value x_0 is the value of the first input variable, A, and the dashed vertical may be used to determine the values of the membership functions $A_1(x_0)$ and $A_2(x_0)$. The same process is applied to the second input variable B, where the value y_0 determines $B_1(y_0)$ and $B_2(y_0)$.

Figure A-4 Example for fuzzy inference



This example uses the intersection (min) operator to combine the membership function values of the antecedents. Thus the weights for both rules are computed as

$$\alpha_1 = \min(A_1(x_0), B_1(y_0)) = B_1(y_0)$$
$$\alpha_2 = \min(A_2(x_0), B_2(y_0)) = A_2(x_0) \tag{EQ 1-5}$$

The output variables C_i are scaled by α_i (product operator). The resulting fuzzy output of the FLC is obtained by combining both scaled output variables using the max-operator. The final result, the output of the FLC, is represented in the rightmost diagram.

Defuzzification

Since an FLC is used to control a physical process, the fuzzy output of the FLC, as shown in Figure A-4, has to be converted into a crisp value, like the position of a valve, the voltage for a motor, etc.

The majorly employed conversion method is the computation of the **center of gravity** (COG) of the resulting fuzzy value:

$$\text{output} = \int \frac{\mu_{\text{consequent}}(x) \cdot x}{\mu_{\text{consequent}}(x)} \qquad \text{(EQ 1-6)}$$